UNIVERSIDAD CARLOS III DE MADRID
Departamento de Teoría de la Señal y Comunicaciones

DOCTORAL THESIS

PARTICLE FILTERS FOR TRACKING IN WIRELESS SENSOR NETWORKS

Author: KATRIN ACHUTEGUI RONCAL
Supervised by: JOAQUÍN MÍGUEZ ARENAS
September 2013

Tesis Doctoral:	PARTICLE FILTERS FOR TRACKING IN WIRELESS SENSOR NETWORKS
Autor:	Katrin Achutegui Roncal
Director:	D. Joaquín Míguez Arenas
Fecha:	

Tribunal

Presidente:

Vocal:

Secretario:

Acknowledgements

I would like to express my deep gratitude to my PhD supervisor, Joaquín Míguez Arenas for his patient guidance, enthusiastic encouragement and useful critiques provided throughout the development of the thesis. There is no doubt that Joaquín is capable of producing great quality research however he is capable of producing something, to my view, much more difficult: researchers.

I would also like to thank Javier Rodas and Carlos Escudero for their contribution with an extremely high quality work that lead to many interesting results and publications. Without their contribution this thesis would not be possible.

I would also like to thank Jesse Read for his important contribution on distributed particle filters, for his encouragement and for the interesting discussions we have had on particle filters, all which have lead to the defense of practical implementations of particle filters as well as the development of many great ideas. Maybe one day we will have enough time to try them all.

I would also like to thank everyone on our research group (all, current and past) for the coffees, lunches and relaxing times that have made work hours much more pleasant.

Finally, I would like to thank Edu for his support and encouragement through my study.

Abstract

The goal of this thesis is the development, implementation and assessment of efficient particle filters (PFs) for various target tracking applications on wireless sensor networks (WSNs).

We first focus on developing efficient models and particle filters for indoor tracking using received signal strength (RSS) in WSNs. RSS is a very appealing type of measurement for indoor tracking because of its availability on many existing communication networks. In particular, most current wireless communication networks (WiFi, ZigBee or even cellular networks) provide radio signal strength (RSS) measurements for each radio transmission. Unfortunately, RSS in indoor scenarios is highly influenced by multipath propagation and, thus, it turns out very hard to adequately model the correspondence between the received power and the transmitter-to-receiver distance. Further, the trajectories that the targets perform in indoor scenarios usually have abrupt changes that result from avoiding walls and furniture and consequently the target dynamics is also difficult to model.

In Chapter 3 we propose a flexible probabilistic scheme that allows the description of different classes of target dynamics and propagation environments through the use of multiple switching models. The resulting state-space structure is termed a generalized switching multiple model (GSMM) system. The drawback of the GSMM system is the increase in the dimension of the system state and, hence, the number of variables that the tracking algorithm has to estimate. In order to handle the added difficulty, we propose two Rao-Blackwellized particle filtering (RBPF) algorithms in which a subset of the state variables is integrated out to improve the tracking accuracy.

As the main drawback of the particle filters is their computational complexity we then move on to investigate how to reduce it via de *distribution of the processing*. Distributed applications of tracking are particularly interesting in situations where high-power centralized hardware cannot be used. For example, in deployments where computational infrastructure and power are not available or where there is no time or trivial way of connecting to it. The large majority of existing contributions related to particle filtering, however, only offer a theoretical perspective or computer simulation studies, owing in part to the complications of real-world deployment and testing on low-power hardware.

In Chapter 4 we investigate the use of the distributed resampling

with non-proportional allocation (DRNA) algorithm in order to obtain a distributed particle filtering (DPF) algorithm. The DRNA algorithm was devised to speed up the computations in particle filtering via the parallelization of the resampling step. The basic assumption is the availability of a set of processors interconnected by a high-speed network, in the manner of state-of-the-art graphical processing unit (GPU) based systems. In a typical WSN, the communications among nodes are subject to various constraints (i.e., transmission capacity, power consumption or error rates), hence the hardware setup is fundamentally different.

We first revisit the standard PF and its combination with the DRNA algorithm, providing a formal description of the methodology. This includes a simple analysis showing that (a) the importance weights are proper and (b) the resampling scheme is unbiased. Then we address the practical implementation of a distributed PF for target tracking, based on the DRNA scheme, that runs in real time over a WSN. For the practical implementation of the methodology on a real-time WSN, we have developed a software and hardware testbed with the required algorithmic and communication modules, working on a network of wireless light-intensity sensors.

The DPF scheme based on the DRNA algorithm guarantees the computation of proper weights and consistent estimators provided that the whole set of observations is available at every time instant at every node. Unfortunately, due to practical communication constraints, the technique described in Chapter 4 may turn out unrealistic for many WSNs of larger size. We thus investigate in Chapter 5 how to relax the communication requirements of the DPF algorithm using (a) a random model for the spread of data over the WSN and (b) methods that enable the out-of-sequence processing of sensor observations. The presented observation spread scheme is flexible and allows tuning of the observation spread over the network via the selection of a parameter. As the observation spread has a direct connection with the precision on the estimation, we have also introduced a methodology that allows the selection of the parameter a priori without the need of performing any kind of experiment. The performance of the proposed scheme is assessed by way of an extensive simulation study.

Resumen

De formal general, el objetivo de esta tesis doctoral es el desarrollo y la aplicación de filtros de partículas (FP) eficientes para diversas aplicaciones de seguimiento de blancos en redes de sensores inalámbricas (*wireless sensor networks* o *WSNs*).

Primero nos centramos en el desarrollo de modelos y filtros de partículas para el seguimiento de blancos en entornos de interiores mediante el uso de medidas de potencia de señal de radio (*received signal strength* o *RSS*) en *WSNs*. Las medidas *RSS* son un tipo de medida muy utilizada debido a su disponibilidad en redes ya implantadas en muchos entornos de interiores. De hecho, en muchas redes de comunicaciones inalámbricas actuales (WiFi, ZigBee o incluso las redes de telefonía móvil), se pueden obtener medidas de *RSS* sin necesidad de modificación alguna. Desafortunadamente, las medidas *RSS* en entornos de interiores suelen distorsionarse debido a la propagación multitrayecto por lo que resulta muy difícil modelar adecuadamente la relación entre la potencia de señal recibida y la distancia entre el transmisor y el receptor. Otra dificultad añadida en el seguimiento de interiores es que las trayectorias realizadas por los blancos suelen tener por lo general cambios muy bruscos y en consecuencia el modelado de las trayectorias dinámicas es una actividad muy compleja.

En el Capítulo 3 se propone un esquema probabilístico flexible que permite la descripción de los diferentes sistemas dinámicos y entornos de propagación mediante el uso de múltiples modelos conmutables entre sí. Este esquema permite la descripción de varios modelos dinámicos y de propagación de forma muy precisa de manera que el filtro sólo tiene que estimar el modelo adecuado a cada instante para poder hacer el seguimiento. El modelo de estado resultante (modelo de conmutación múltiple generalizado, *generalized switiching multiple model* o *GSMM*) tiene el inconveniente del aumento de la dimensión del estado del sistema y, por lo tanto, el número de variables que el algoritmo de seguimiento tiene que estimar. Para superar esta dificultad, se proponen varios algoritmos de filtros de partículas con reducción de la varianza (*Rao-Blackwellized particle filtering (RBPF) algorithms*) en el que un subconjunto de las variables de estado, incluyendo las variables indicadoras de observación, se integran a fin de mejorar la precisión de seguimiento.

Dado que la mayor desventaja de los filtros de partículas es su complejidad computacional, a continuación investigamos cómo reducirla distribuyendo el procesado entre los diferentes nodos de la red. Las

aplicaciones distribuidas de seguimiento en redes de sensores son de especial interés en muchas implementaciones reales, por ejemplo: cuando el hardware usado no tiene suficiente capacidad computacional, si se quiere alargar la vida de la red usando menos energía, o cuando no hay tiempo (o medios) para conectarse a la toda la red. La reducción de complejidad también es interesante cuando la red es tan extensa que el uso de hardware con alta capacidad de procesamiento la haría excesivamente costosa.

La mayoría de las contribuciones existentes ofrecen exclusivamente una perspectiva teórica o muestran resultados sintéticos o simulados, debido en parte a las complicaciones asociadas a la implementación de los algoritmos y de las pruebas en un hardware con nodos de baja capacidad computacional. En el Capítulo 4 se investiga el uso del algoritmo *distributed resampling with non proportional allocation* (*DRNA*) a fin de obtener un filtro de partículas distribuido (FPD) para su implementación en una red de sensores real con nodos de *baja capacidad computacional*. El algoritmo *DRNA* fue elaborado para acelerar el cómputo del filtro de partículas centrándose en la paralelización de uno de sus pasos: el remuestreo. Para ello el *DRNA* asume la disponibilidad de un conjunto de procesadores interconectados por una red de alta velocidad.

En una red de sensores inalábrica, las comunicaciones entre los nodos suelen tener restricciones (debido a la capacidad de transmisión, el consumo de energía o de las tasas de error), y en consecuencia, la configuración de hardware es fundamentalmente diferente. En este trabajo abordamos el problema de la aplicación del algoritmo de *DRNA* en una *WSN* real. En primer lugar, revisamos el FP estándar y su combinación con el algoritmo *DRNA*, proporcionando una descripción formal de la metodología. Esto incluye un análisis que demuestra que (a) los pesos se calculan de forma adecuada y (b) que el paso del remuestreo no introduce ningún sesgo. A continuación describimos la aplicación práctica de un FP distribuído para seguimiento de objetivos, basado en el esquema *DRNA*, que se ejecuta en tiempo real a través de una *WSN*. Hemos desarrollado un banco de pruebas de software y hardware donde hemos usado unos nodos con sensores que miden intensidad de la luz y que a su vez tienen una capacidad de procesamiento y de comunicaciones limitada. Evaluamos el rendimiento del sistema de seguimiento en términos de error de la trayectoria estimada usando los datos sintéticos y evaluamos la capacidad computacional con datos reales.

El filtro de partículas distribuído basado en el algoritmo *DRNA* garantiza el cómputo correcto de los pesos y los estimadores a condición de que el conjunto completo de observaciones estén disponibles en cada instante

de tiempo y en cada nodo. Debido a limitaciones de comunicación esta metodología puede resultar poco realista para su implementación en muchas redes de sensores inalámbricas de tamaño grande. Por ello, en el Capítulo 5 investigamos cómo reducir los requisitos de comunicación del algoritmo anterior mediante (a) el uso de un modelo aleatorio para la difusión de datos de observación a través de las red y (b) la adaptación de los filtros para permitir el procesamiento de observaciones que lleguen fuera de secuencia. El esquema presentado permite reducir la carga de comunicaciones en la red a cambio de una reducción en la precisión del algoritmo mediante la selección de un parámetro de diseño. También presentamos una metodología que permite la selección de dicho parámetro que controla la difusión de las observaciones a priori sin la necesidad de llevar a cabo ningún tipo de experimento. El rendimiento del esquema propuesto ha sido evaluado mediante un estudio extensivo de simulaciones.

Contents

1 Introduction **1**

1.1 Wireless sensor networks 1
1.2 Localization in WSNs 3
1.2.1 Physical measurements 4
1.2.2 Positioning 4
1.2.3 Navigation and tracking 5
1.2.4 Bayesian filters 6
1.3 Contributions 7
1.3.1 Indoor tracking with RSS 8
1.3.2 Tracking with distributed particle filtering 8
1.4 Thesis organization 9

2 Particle filters for target tracking **11**

2.1 Notation 11
2.2 Bayesian filtering for target tracking 12
2.2.1 State-space model representation 12
2.2.2 Bayesian filtering 14
2.2.3 The Kalman filter and its extensions 16
2.3 Nonlinear filtering via sequential importance sampling 23
2.3.1 Exact Monte Carlo sampling 23
2.3.2 Importance sampling 25
2.3.3 Sequential importance sampling: particle filtering . . . 27
2.4 Improved particle filters 30
2.4.1 SIS with optimal importance function 30
2.4.2 The auxiliary particle filter 32
2.4.3 The Rao-Blackwellized particle filter 33
2.4.4 Summary 36

3 A multi-model sequential Monte Carlo methodology for indoor tracking **39**
3.1 Introduction . . . 40
3.2 System model . . . 43
3.2.1 Motion models . . . 43
3.2.2 Measurement models . . . 45
3.2.3 Summary . . . 46
3.3 Construction of observation models . . . 46
3.3.1 Experimental scenario and data . . . 46
3.3.2 Polynomial observation sub-models . . . 48
3.3.3 Logarithmic path-loss observation sub-models . . . 51
3.4 Tracking algorithms . . . 54
3.4.1 A Rao-Blackwellized particle filter for multiple models 54
3.4.2 Evaluation of the weights . . . 57
3.4.3 Importance functions . . . 58
3.4.4 An auxiliary particle filter for multiple models . . . 58
3.4.5 Computational complexity of the algorithms . . . 60
3.5 Computer simulations results . . . 62
3.5.1 Order of polynomial models . . . 63
3.5.2 Tracking performance . . . 66
3.5.3 Comparison with the interacting multiple model methodology . . . 69
3.5.4 Gain from using multiple models . . . 71
3.6 Experimental results . . . 71
3.7 Conclusions . . . 72

4 A distributed particle filter implementation **77**
4.1 Introduction . . . 77
4.2 System model . . . 79
4.2.1 Motion model . . . 79
4.2.2 Measurement model . . . 81
4.3 Experimental set-up and observation models . . . 82
4.4 Distributed particle filtering . . . 87
4.4.1 General structure . . . 87
4.4.2 Particle exchange . . . 88
4.4.3 Local processing . . . 89
4.4.4 Estimation . . . 90
4.4.5 Summary . . . 92
4.5 Simulations and experimental results . . . 92
4.5.1 Setup . . . 92

4.5.2 Computer simulations 94
4.5.3 Experimental results 97
4.5.4 Performance study 97
4.5.5 Limitations and remarks 101
4.6 Conclusions 101

5 A Distributed particle filter for wireless sensor networks with stochastic observation exchange 103
5.1 Introduction 104
5.2 System model 104
5.2.1 Motion model 105
5.2.2 Measurement model 105
5.3 Distributed tracking algorithm 106
5.3.1 General structure 106
5.3.2 Observation spread model 107
5.3.3 Weight update 112
5.3.4 Particle exchange step 112
5.3.5 Estimation 116
5.3.6 Summary of the DPF scheme 1 (DPF-1) 117
5.3.7 Summary of the DPF scheme 2 (DPF-2) 117
5.4 Analysis 119
5.4.1 Out-of-sequence measurement handling 119
5.4.2 Propagation of observation data 121
5.5 Simulation results 123
5.5.1 Target prior parameters and example 124
5.5.2 Network connectivity and Markov chain parameter selection 125
5.5.3 Effect of the number of processing elements 126
5.5.4 Synchronized versus non-synchronized particle exchange 127
5.5.5 Effect of the number of total jumps 130
5.5.6 Smoothed estimates 131
5.5.7 Effect of the number of intermediate jumps 132
5.5.8 Limitations and remarks 135
5.6 Conclusions 137

6 Summary and Conclusions 139
6.1 Summary 139
6.1.1 Indoor tracking with RSS measurements 139
6.1.2 A distributed particle filter implementation on a WSN 141

6.1.3 Distributed particle filtering on a WSN with random spread of the measurement data 142
6.2 Future research 143
6.2.1 Multiple targets 143
6.2.2 Convergence results 144
6.2.3 Suboptimal DPF schemes 145

A Recursive computation of the prior density **147**

B Derivation of the likelihood **149**

C Acronyms and abbreviations **151**

D Notation **155**

References **156**

List of Tables

2.1 Kalman filter (KF) 18
2.2 Extended Kalman filter (EKF) 20
2.3 Unscented Kalman filter (UKF) 24
2.4 Sequential importance sampling (SIS) 28
2.5 Sequential importance resampling (SIR) with a prior importance function. 31
2.6 Auxiliary particle filter for SIR (A-SIR) 34
2.7 Rao-Blackwellized particle filter. 37

3.1 Number of observations and number of different distances to which we collected RSS data in order to construct the polynomial and the logarithmic models. 50
3.2 Rao-Blackwellized particle filter for the GSMM system. . . . 59
3.3 Auxiliary RBPF algorithm for the GSMM system. 61
3.4 Algorithm and parameters table for the RBPF and A-RBPF algorithms. 64
3.5 Mean absolute error of the position for different ordered polynomial observation models 65
3.6 Mean absolute error of the position for Gaussian likelihood functions 68
3.7 Mean absolute error of the position for truncated Gaussian likelihood functions 68
3.8 Mean absolute error of the position for logarithmic path-loss models 76

4.1 Distributed particle filtering (DPF) algorithm. 93
4.2 DPF and model parameters. 94

4.3 The processing time (seconds) per timestep of the DPF algorithm for various values of N. The total number of particles is constant ($M = 100$); $100/N$ per PE. Note that this time does not include network activity. 99

4.4 The *network activity* (in terms of packets and bytes) per timestep for J motes comprised of N PEs and $J - N$ SEs. We a store each 4-dimensional state $\mathbf{y}_t$ with its weight w_t in 20 Bytes (4 Bytes for each number) and each observation y_t in 1 Byte. 100

4.5 Memory usage for M particles, a state size of $d = 4$, and N PEs. Assuming 4 Bytes to store floating point values (as is the case on the iMote2). 100

5.1 Random spread of the observation data for the proposed DPF scheme. 111

5.2 Synchronized particle exchange process for the random spread DPF. 115

5.3 Random spread DPF scheme (DPF-1). 118

5.4 Random spread DPF scheme without synchronized particle exchange (DPF-2). 120

5.5 Mean absolute error of the position in meters for the CPF and the DRNA scheme 127

5.6 Average mean absolute error (MAE) and standard deviation of the absolute error (SDE) of the DPF-2 algorithm for different types of jumps. The %iMAE and %iSDE indicate the percentage increase in the mean and standard deviation of the errors *with respect to* the DRNA. 131

5.7 Average mean absolute error (MAE) and standard deviation of the absolute error (SDE) of the DPF-2 algorithm for a total of $B \approx 68$ jumps and a range of intermediate jumps $L \approx B/4, B/3, B/2, B$. The %iMAE and %iSDAE indicate the percentage increase in the mean and standard deviation of the errors *with respect to* $L = B$, that is, when the total jumps are performed in one time instant. 133

5.8 Average mean absolute error (MAE) and standard deviation of the absolute error (SDE) of the DPF-2 algorithm for a total of $B = 120$ jumps and a range of intermediate jumps $L = B/4, B/3, B/2, B$. The %iMAE and %iSDAE indicate the percentage increase in the mean and standard deviation of the errors *with respect to* $L = B$, that is, when the total jumps are performed in one time instant. 134

List of Figures

3.1 Indoor wireless sensor network scenario. The plot in the left shows the deployment of $J = 9$ sensors and their positions in the 6×10 meter area and the plot in the right shows the positions of the mobile sensor when taking static measurements. 49

3.2 Raw RSS data collected by the mobile sensor from anchor sensors 1, 2, 3, 4, 5 and 6. The x-axis shows the transmitter-receiver distance and the y-axis shows the RSS measured in dBs. The title of each graph displays the specific sensor identifier, $j = 1, 2, 3, 4, 5, 6$, and its position in the 6×10 meter area. Note the difference in the transmitted power from the two sensor types: sensors $1, 3, 5$ are XBee-Pro and sensors $2, 4, 6$ are XBee. 49

3.3 Raw RSS data collected from sensors 1, 2, 3, 4, 5 and 6 for the logarithmic models. The x-axis shows the distance among nodes and the y-axis shows the RSS measured in dBs. The title of each graph displays the specific sensor identifier, $j = 1, 2, 3, 4, 5, 6$, and its position in the 6×10 meter area. Note the difference in the transmitted power from the two sensor types: sensors $1, 3, 5$ are XBee-Pro and sensors $2, 4, 6$ are XBee. 50

3.4 Mean (solid line) and standard deviation (dashed line) of sensor 1. The scattered dots represent the raw data. The fitted mean comes from polynomials $g_1^1(d)$ and $g_2^1(d)$ and the standard deviation has been taken from from polynomials $h_1^1(d)$ and $h_2^1(d)$. Figure (a) corresponds to model $m_{1,t} = 1$ and polynomial order $n = 3$, Figure (b) corresponds to model $m_{1,t} = 2$ and polynomial order $n = 3$, Figure (c) corresponds to model $m_{1,t} = 1$ and polynomial order $n = 5$, Figure (d) corresponds to model $m_{1,t} = 2$ and polynomial order $n = 5$, Figure (e) corresponds to model $m_{1,t} = 1$ and polynomial order $n = 7$, Figure (f) corresponds to model $m_{1,t} = 2$ and polynomial order $n = 7$. 52

3.5 Mean and standard deviation of the logarithmic observation model. Figure (a) corresponds to model $m = 1$ and Figure (b) to model $m = 2$. We use the same pair of models for every sensor in the network. 59

3.6 Average execution time per time step of the RBPF and the A-RBPF algorithms. 64

3.7 Results of target tracking with three different reference trajectories and four types of synthetic observation data, each one created according to the four likelihood functions described in Section 3.3.1. The simulated trajectories are plotted in solid black lines, the estimated trajectories in solid red lines and the sensors are depicted with dark squares. To perform tracking we have used four different algorithms that correspond to an A-RBPF algorithm of 200 particles that uses the four different likelihood functions. In each column we show the tracking performance of a different algorithm and in each row we show the tracking capabilities of the algorithms for a specific trajectory. 67

3.8 Tracking performance of the A-RBPF algorithm with experimental data: results of target tracking with three different reference trajectories and real RSS observations collected in the experimental setup described in Section 3.3.1. The real trajectories are plotted in solid black lines, the estimated trajectories in solid red lines and the sensors are depicted with dark squares. To perform tracking we have used four different algorithms that correspond to an A-RBPF scheme that uses the four different likelihood functions we introduced. In each column we show the tracking performance of a different algorithm and in each row we show the tracking capabilities of the algorithms for a specific trajectory. 73

3.9 Tracking performance of the A-RBPF algorithm of 200 particles and the IMM-UKF with experimental data. The real trajectories are plotted in solid black lines, the estimated trajectories in solid red lines and the sensors are depicted with dark squares. All the algorithms use logarithmic observation models with Gaussian likelihoods. The column in the left shows the outcome of the IMM-UKF tracker and the column in the right illustrates the performance of the A-RBPF algorithm. 74

4.1 Tracking scenario of 3.2×6.0 meters (a bird's-eye view). The bold line indicates the light source (a window). There are $J = 10$ motes equipped with light sensors are arranged around the edges, indicated by squares. The entry to the scenario lies at the bottom-right corner. 83

4.2 The detection zone of a sensor, labeled Z in the plot, is the area enclosed by a triangle with one vertex at the sensor and the other two vertices at the sides of the light source. 84

4.3 The solid line represents the light signal measured when the target (a walking person) moves randomly outside the detection zone. The dashed line is the signal when the target moves randomly inside the detection zone. Rather than a reduction in the light level, the presence of a target within the detection zone causes a large variance in the sensor readings. 85

4.4 Histogram of the position error in meters for both the distributed and centralized versions of the PF (both with a total of 100 particles) over 100 simulated trajectories. The plot in the left corresponds to the results of a CPF whilst the plot in the right displays the results obtained with the DPF algorithm. In both cases the error is about half a meter on average. 95
4.5 The simulated (black) paths for for simulations 1–4, and the corresponding DPF-estimated paths (red); each over $T = 18$ time steps. 96
4.6 Results of the DPF algorithm (solid line, $N = 4$) tracking a target walking a prescribed trajectory (dashed line). Of the three straight lines making up the true path, each is walked slightly faster than the previous one, with a pause of about one second taken at the point where the direction is changed (indicated by hollow circles). The path is walked in $T = 18$ timesteps (18 seconds in our setup). 98

5.1 Example of two dimensional target trajectory and associated observation 124
5.2 Network topology and observation spread probability. 126
5.3 Mean absolute error of the position in meters for the DPF-1 and the DPF-2 for a range of number of total jumps. 128
5.4 Average transmitting PEs per time instant in the particle exchange step for the DPF-1. 128
5.5 Symmetric difference and normalized symmetric difference between the sets of observations of sensor $n = 7$ and its neighbors. 130
5.6 Average MAE of the position for the CPF, DRNA and DPF-2 obtained with smoothed estimates. The estimation has been performed at estimation instants of $t - k$ where $k = 0, 1, 2, ..., 20$ with observation up to t. 132
5.7 Average MAE of the position for the DPF-2 obtained with smoothed estimates. These experiments have been performed for a fixed number of total retransmissions of $B = 68$ (left) and $B = 180$ (right). We have used $K = 100$ particles per PE. 135

Chapter 1

Introduction

1.1 Wireless sensor networks

Wireless sensor networks (WSNs) are one of the most extensively researched technologies of the past decade [59, 112]. The features that make them so attractive is that they can perform monitoring tasks in an autonomous and cheap manner for a long period of time compared to other technologies. By definition a WSN is a group of spatially distributed nodes that have sensing and computing capabilities. Its main goal is to monitor the physical environment around it and, to do so, sensors may measure different physical or environmental conditions (temperature, pressure, light, acceleration, sound, humidity, images, etc.). Once sensors have taken measurements they communicate with each other via a wireless technology (e.g. bluetooth, ZigBee or WiFi) and then they fuse all the information. Due to the diversity of types of physical measurements they can collect and process, there is a wide variety of applications in which WSNs are used: military surveillance, infrastructure security, environmental and habitat monitoring, industrial sensing, traffic control, etc. [59].

As new applications arise and new, usually smaller, sensors are developed engineers keep on searching for cheaper, faster and more robust WSNs. Specifically, researchers are focusing on

- increasing the WSN life via reducing the battery usage of the nodes [58, 107, 52],
- reducing the communication load in order to improve the reliability of the network [119, 54, 82],

- increasing the reliability and robustness to node failures [59, 61, 43] and

- reducing the size of the sensors and, thus, reducing costs [59, 119].

These trends (longer living, cheaper and more robust WSNs with reliable communications) lead to algorithms that have specific, and sometimes conflicting, requirements. For example, if we want the batteries of the nodes to last for a longer period of time we have to design algorithms that have a low computational complexity, where each node hardly makes any processing and focuses on transmitting the information to a central node that performs the main part of the computation. However, if we want to improve robustness towards single node failure we need to avoid the dependence on a central node, and we hence aim at distributing the processing over the network. This setting, unfortunately, increases the communication load in the WSN. Overall, the need for new algorithms that meet some of these specific requirements has increased over the past few years.

In sensor networks, accurate localization of wireless devices is a crucial requirement for many emerging location-aware systems [93, 103]. Fields of applications include search and rescue, medical care, intelligent transportation, location-based billing, security, home automation, industrial monitoring and control, location-assisted gaming, and social networking [106, 74]. Further, many applications of WSNs require knowledge of the position of objects or persons of interest, often termed "targets". One example of a "target tracking" application is the update of inventory in supermarkets. Stores such as Wal-Mart have implemented RFID systems in their warehouses in such a way that all items from various manufacturers being stored have a tag attached to them [115]. Sensors discretely attached to walls or embedded in doors and ceilings, track the location history and use of items. The sensor network can automatically locate items, report on those needing servicing and report unexpected large-scale movements of items or significant changes in inventory levels. Some systems today (for example, those based on bar-codes) provide inventory tracking; full sensor-net based systems will eliminate manual scanning and provide more data than simply location.

It has also been suggested [96] that planning trip itineraries could become very simple if every vehicle in a large city has one or more attached sensors. These sensors capable of detecting their location, traffic speed and densities, road conditions and so on could exchange information summaries such as which roads are highly congested and whether any alternate routes should

be chosen. We can also know about the driving conditions or pollution levels and routes could be planned based on any social activities in the area as well.

During the last few years, there have been intensive research activities in the field of WSN-based positioning and tracking and various solutions have been investigated [111, 113]. The main trend now is toward the integration of heterogeneous technologies to ensure global coverage and high accuracy in all possible scenarios, leading to a seamless localization system available anywhere anytime. This is often known as the localization problem in sensor networks [93]. While satellite-based navigation is well consolidated for open sky scenarios, localization in harsh environments (e.g., indoor or in urban canyons) is still an open issue that requires the use of complementary WSN technologies.

1.2 Localization in WSNs

Localization can be performed on static or moving objects. Usually, localization performed on static objects is referred to as positioning whilst tracking and navigation are terms associated to localization of moving objects. The difference between tracking and navigation is purely conceptual. In navigation, the goal is to estimate the position and velocity of one's own platform or vehicle, while in tracking, the position and velocity of some other object or person is to be inferred.

The localization methodology changes drastically depending on the environment (either indoor or outdoor), essentially due to the types of technologies available in each case. For example, in outdoor scenarios very often one can solve the localization problem with a satellite positioning system (e.g. GPS) however this technology is not yet available in indoor scenarios despite some recent efforts [73, 88, 21]. On the other hand, in indoor scenarios it is easier to deploy WiFi or bluetooth nodes that are not readily available in many outdoor settings. Additionally, the same technology may behave differently depending on the environment. For example, ZigBee motes[1] have a specific radio signal strength decay in outdoor environments, while in indoor scenarios the rays ricochet off the walls and furniture creating reflected and diffracted versions [92, 46]. Due to these reasons, indoor and outdoor localization are research areas of their own and the type of algorithms one can use in each environment varies greatly.

[1] A "mote" is understood here as a network node with combined sensing, communication and processing capability.

In the sequel we briefly review the usual types of position-dependent data that can be collected with various sensing technologies and then provide a quick overview of positioning, navigation and tracking algorithms.

1.2.1 Physical measurements

Most of the physical measurements used for localization can be categorized in two types: range measurements and bearing measurements. Localization with range measurements is performed via the principle of trilateration and localization with bearing measurements via triangulation [48].

Active radar and sonar sensors emit a pulse and measure the round trip time for the returned pulse, which can be converted into range information. This measurement is known as time of arrival (ToA). Passive radar and sonar systems listen to the target's own emission and then only the difference between two pairs of sensors can be computed. This type of measurement is known as time difference of arrival (TDoA). The power of radio, acoustic, magnetic and seismic waves decays exponentially with distance, which also provides a kind of range measurement. This type of measurement is denoted received signal strength or RSS. Radar and similar sensors with directional antennas assign bearing to detected targets. Vision sensors such as cameras can provide two angles to objects, but no range data. Antenna arrays and lobe forming techniques also give bearing in the form of angle of arrival (AoA) information. Other sensors can measure the Doppler shift of signal frequencies in order to estimate the relative velocity of a moving object with respect to the sensor platform [33].

Among these measurement types, RSS is a specially popular type of data because we can use the radio transmitter of the mote as a sensor itself. Furthermore, other elements containing radio transmitters of other types, such as WiFi, ZigBee or bluetooth, can become part of the WSN as required. The behavior of the different types of RSS measurements, however, depends on the specific technology.

The type of physical measurements available, and their mathematical description, is very important as it strongly influences the type of localization algorithm we can use.

1.2.2 Positioning

The positioning problem is that of estimating the position of a static object given some measurements related to it. If we know the relationship between the obtained measurement and the target (that is, if we know

the measurement function) and possibly the statistics of the observation noise, the positioning problem reduces to a classical estimation problem. In such a situation there are a variety of estimation methods that can solve this problem. Some of the most used techniques are least squares (LS), maximum likelihood (ML), nonlinear least squares (NLS), weighted least squares (WLS), separable least squares (SLS), etc. [48].

Depending on the type of measurement function, namely, depending on whether it is linear or nonlinear with respect to the position of the target, the estimation problem may become hard. For example, in [83] the authors pose the problem of positioning with a WSN as a soft convex feasibility problem and they consider the well-known projection onto convex sets technique to solve it. In [69] they propose a joint communication and positioning system based on ML channel parameter estimation. The parameters of the physical channel, needed for positioning, and the channel coefficients of the equivalent discrete-time channel model, needed for communication, are estimated jointly using a priori information about pulse shaping and receive filtering. In [5], they propose two (ToA) estimators for ultra-wideband (UWB) signals based on the phase difference between the discrete Fourier transforms (DFT) of the transmitted and received signals

1.2.3 Navigation and tracking

In navigation and target tracking problems, the goal is to keep estimating the position and velocity of a moving object over time. For such problems it is most natural to represent the physical system of interest by means of a dynamic model [12, 99] that describes, in a quantitative manner,

- how the target moves over the region monitored by the WSN and
- how the measurements collected by the WSN relate to the target position and/or velocity.

A popular class of models that can appropriately provide this description is that of random state-space models. State space models consist of a set of differential or difference equations (depending on whether a continuous or discrete time representation is sought) that describes a physical system.This type of representation provides a convenient and compact way to model and analyze systems with multiple inputs and outputs. Each model consists of two parts: the state process and the observation process. The state process is thought of as a hidden underlying stochastic process and its evolution over time is described by a state equation. The observation process

is supposed to be accessible and is related to the state process via the measurement or observation equation. In tracking applications the state of the system contains all the relevant kinematic information regarding the target (position, velocity, heading, etc.) and the observation process models the measurements collected by the WSN.

Given the state equation and measurement equation in state-space form, we are faced with a computational inference problem where we wish to estimate the state process from the sequence of measurements obtained from it. Furthermore, we wish to estimate the state at each time instant, given the measurements up to and including that time instant. Due to the amount of data that has to be processed, we aim at a sequential and recursive solution, so that we do not have to recompute everything from the very beginning at each time step.

Filtering (or stochastic filtering) is precisely the operation of extracting the information about the systems state at a particular time instant by using data up to that time instant. Specifically, it attempts to construct estimators of the state conditional on the sequence of received measurements. There are different types of filters that can solve this problem. The H_∞ filter can perform filtering without any assumption on the noise distribution [8]. Gaussian and recursive least squares (RLS) filters have also been widely used [48]. We focus on Bayesian filters [44] in the sequel, because they provide a solution that is often computationally appealing and mathematically consistent for state-space models.

1.2.4 Bayesian filters

Given a prior probability distribution for the state process and a state-space model, the optimal Bayesian filter computes the posterior distribution of the state given the observations collected up to a given time instant [10]. Unfortunately, such computation can only be carried out exactly in a few exceptional cases, including when the model is linear Gaussian (and the solution is given by the well known Kalman filter [116]) or the state space is discrete and finite.

In most practical situations however we are faced with highly nonlinear, non-Gaussian and non-stationary problems with continuous valued target states. As a consequence we cannot compute the optimal Bayesian filter and instead we are forced to use approximations. One popular approach consists in approximating the state-space equations with linear functions in order to later on apply the Kalman filter (KF). This particular approach is called the extended Kalman filter (EKF) [3]. Several other variants of the standard KF

have been proposed, including the popular unscented Kalman filter (UKF) [65]. This filter that approximates the posterior pdf as a Gaussian which, in turn, is represent by a set of deterministically chosen set points. The idea behind the UKF is to propagate these points through the appropriate non linear function rather than performing an analytical approximation to the state-space model.

Another approach consists in the discretization of the possible target values. This, in a sense, is a numerical approximation of the Bayesian integral. The methods following this approach are called grid based methods [99]. Another possible approach is to approximate the posterior pdf as a mixture of Gaussians. This methods, called the Gaussian sum filters [3], can pose severe difficulties if the number of Gaussians of the mixture is unknown.

Over the past twenty years particle filters (PFs) [40, 41, 47, 77, 99, 35, 22] have attracted interest from practitioners as an approximation to the optimal Bayesian filter for target tracking [99, 7]. PFs perform sequential Monte Carlo (SMC) estimation based on point mass representations of the probability distribution of interest. The basic principle of the PFs is the sequential application of the importance sampling methodology. Their main advantage is their capability of representing nearly any probability distribution and thus of handling any state-space model while their disadvantage lays in their computational cost. However with the advent of more powerful computers and the development of more efficient particle filtering algorithms they now have earned an important place in the tracking community (for example, there is an open source Matlab toolbox called PF Toolbox that includes the SIR and SIS particle filters, as well as the extended Kalman filter).

1.3 Contributions

This thesis is mainly concerned with two target tracking problems. The first one is the problem of indoor tracking using received signal strength (RSS) as a position-dependent data measurement. The second one is the problem of devising a distributed particle filtering algorithm for tracking in WSNs. Specifically, the main contributions of this work are the following.

1. A class of state-space models suitable for target tracking applications in highly structured physical environments and a

2. family of efficient PF algorithms for this class of models.

3. Specific models fitted with experimental RSS indoor data and experimental performance results for real-world time series.

4. An experimental demonstration of the applicability of a class of distributed PFs on a WSN environment for target tracking with binary data.

5. A novel distributed PF methodology with an stochastic mechanism for data sharing among the nodes of a WSN.

1.3.1 Indoor tracking with RSS

Even though RSS is a very popular type of data for tracking, it is highly influenced by multipath propagation and, therefore, it turns out very hard to adequately model the correspondence between the received power and the transmitter-to-receiver distance. Although various models have been proposed in the literature, they often require the use of very large collections of data in order to fit them and display great sensitivity to changes in the radio propagation environment. We advocate the use of switching multiple models that account for different classes of target dynamics and propagation environments and propose a flexible probabilistic switching scheme. The resulting state-space structure is termed a generalized switching multiple model (GSMM) system. Within this framework, we investigate two types of models for the RSS data: polynomial models and classical logarithmic path-loss representations. The first model is more accurate, however it demands an offline model-fitting step. The second one is less precise but it can be fitted using a simple online procedure.

We have designed two tracking algorithms built around a Rao-Blackwellized particle filter, tailored to the GSMM structure, and assessed their performance both with synthetic and experimental measurements.

1.3.2 Tracking with distributed particle filtering

We also address the design of a particle filter (PF) that can be implemented in a distributed manner over a network of wireless sensor nodes, each of them collecting their own local data. This is a problem that has received considerable attention lately and several methods based on consensus, the transmission of likelihood information [56], the truncation and/or the quantization of data have been proposed [30]. However, all existing schemes suffer from limitations related either to the amount of required communications among the nodes or to the accuracy of the filter outputs.

In this work we propose a mathematically sound distributed particle filter for tracking in a real- world indoor wireless sensor network comprised of low-power nodes. The algorithm is built around the distributed resampling with non-proportional allocation (DRNA) algorithm [15] that guarantees the properness of the particle approximations produced by the filter. We present the results of both real-world experiments and computer simulations that use models fitted with real data. In particular, we have carried out real-world tests that show how the proposed distributed particle filter can successfully track a moving target using a network of passive sensors (that provide measurements of ambient light intensity). Additionally, a series of computer simulations run with experimentally-fitted models show a precision of around half a meter for the same scenario.

Even though the DRNA method has been shown to be both efficient and accurate when compared with centralized PFs, it places stringent demands on the communications among nodes that may turn out impractical for large WSNs. We investigate how to reduce this communication load by using (i) a random process for the spread of data over the WSN and (ii) methods that enable the out-of-sequence processing of sensor observations. A numerical illustration of the performance of the new algorithm compared with a centralized PF is provided.

1.4 Thesis organization

This thesis is divided in six chapters, being Chapter 1 this Introduction:

Chapter 2 provides the necessary mathematical background required to understand the rest of the thesis and it has three sections. The first section poses the tracking problem, from a Bayesian point of view, describing its solution for the linear filtering problem. The following section describes the nonlinear tracking Monte Carlo numerical integration methodology and its sequential version, i.e., the PF. The third section describes some of the refined PF versions, which are used and modified in the rest of the thesis.

Chapter 3 is devoted to the engineering of system models and particle filters that improve the tracking with RSS in indoor scenarios with multipath effects. It is divided into six sections. We first introduce the generalized switching multiple model as a general state space model proposed to describe the behavior of both the target dynamics and the RSS observations in indoor scenarios. Later on we devote a section to the fitting of observation functions and noise variances to real experimental data. We then move on to describe possible particle filtering algorithms that can both handle and exploit the

characteristics of the system model. The next sections provide a study based on simulated and experimental results and there is a final section with some concluding remarks.

Chapter 4 focuses on the design of a distributed particle filter for tracking in WSNs and is divided into four sections. The first section describes the system model of the tracking problem. The second section describes the distributed resampling with non-proportional algorithm (DRNA) and gives details on how it can be implemented as a fully decentralized PF for tracking on a WSN. Subsequently, we show some simulated and experimental results and conclude with a discussion of the advantages and disadvantages of the method.

Chapter 5 is focused on a distributed algorithm that relaxes the computational requirements of the DRNA procedure and is divided into four sections. The first section describes the system model of the tracking problem. The second section presents the random observation-spread procedure that we propose in order to alleviate the communication load implied by the DRNA scheme. We further describe the resulting DRNA algorithm with the required modifications that handle the new communication model. The next sections are devoted to a mathematical analysis of the resulting algorithm in terms of communication propagation speed and accuracy. Finally, we provide a numerical study with synthetic data and some concluding remarks.

Chapter 6 summarizes the thesis contributions and presents some possible future research lines.

Chapter 2

Particle filters for target tracking

In this chapter we present some background material needed to understand the rest of this work. In particular, we describe the solution to the target tracking problem from a Bayesian filtering point of view. This, in turn, takes us to the basic particle filter. We also describe some improved particle filters that will be later used and modified for our specific applications.

2.1 Notation

Scalar magnitudes are denoted using regular face letters, e.g., x, while matrices and vectors are written as bold-face upper-case and lower-case letters, respectively, e.g., matrix $\mathbf{X}$ and vector $\mathbf{x}$. We use letter p to denote the true probability density function (pdf) of a random variable or vector. This is an argument-wise notation, common in Bayesian analysis. For two random variables x and y, $p(x)$ is the true pdf of x and $p(y)$ is the true pdf of y, possibly different. The conditional pdf of x given y is written $p(x|y)$. Exactly the same notation is used for probability mass functions (pmf's). Note that a pmf can be formally handled in the same way as a pdf by constructing it as a train of Dirac delta functions, denoted $\delta(.)$, centered at the points where the probability masses are located.

2.2 Bayesian filtering for target tracking

2.2.1 State-space model representation

Using state-space notation we represent all the relevant information regarding the moving target at time instant $t \in \mathbb{N}$ as a real-valued random vector, $\mathbf{x}_t \in \mathbb{R}^{n_x}$, of dimension n_x (note though that complex representation is also possible). The measurements collected from the sensors at time instant t are represented with vector $\mathbf{y}_t \in \mathbb{R}^{n_y}$, where n_y is the dimension of the measurement vector. We use notation $\mathbf{x}_{0:t}$ and $\mathbf{y}_{1:t}$ to refer to the sequence of state vectors from time 0 to time t and the sequence of measurement vectors from time 1 to time t respectively. We then model $\mathbf{x}_{0:t}$ and $\mathbf{y}_{1:t}$ as two Markov stochastic processes in discrete time, namely,

$$\mathbf{x}_t = \mathbf{h}_t(\mathbf{x}_{t-1}, \mathbf{u}_t), \tag{2.1}$$

$$\mathbf{y}_t = \mathbf{g}_t(\mathbf{x}_t, \boldsymbol{\varepsilon}_t), \tag{2.2}$$

where $\mathbf{h}_t : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ and $\mathbf{g}_t : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ are two vector-valued functions which can be either fixed in time or time varying, $\mathbf{u}_t$ represents the dynamic noise of the process and $\boldsymbol{\varepsilon}_t$ represents the measurement noise. Correspondingly, $\mathbf{h}_t$ is often referred to as the state transition function and $\mathbf{g}_t$ as the observation or measurement function.

Depending on the type of target we are tracking (vehicle, person, robot, etc.) and depending on the type of measurements we are handling (RSS, AoA, TDOA, etc.) we design $\mathbf{h}_t$ and $\mathbf{g}_t$ so that they describe both the type of movement of the target and the related measurements as accurately as possible. In the sequel, we describe two commonly used models for target tracking.

Example 1: Constant velocity model with received signal strength (RSS) measurements.

One popular dynamic function is the so-called constant velocity (CV) model [11]. This function represents the movement of a target in a two dimensional region. The velocity is modeled as a discrete-time random walk, which amounts to assuming that it is piece-wise constant in continuous time (hence the name of the model). The target state vector contains the position, $\mathbf{r}_t = [r_{1,t}, r_{2,t}]^\top$, and the velocity, $\mathbf{v}_t = [v_{1,t}, v_{2,t}]^\top$, in a two dimensional region, hence we define the state vector as $\mathbf{x}_t = [\mathbf{r}_t^\top, \mathbf{v}_t^\top]^\top$, and the state

equation is defined as

$$\underbrace{\begin{bmatrix} r_{1,t} \\ r_{2,t} \\ v_{1,t} \\ v_{2,t} \end{bmatrix}}_{\mathbf{x}_t} = \underbrace{\begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} r_{1,t-1} \\ r_{2,t-1} \\ v_{1,t-1} \\ v_{2,t-1} \end{bmatrix}}_{\mathbf{x}_{t-1}} + \underbrace{\begin{bmatrix} \frac{1}{2}T^2 & 0 & 0 & 0 \\ 0 & \frac{1}{2}T^2 & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix} u_{1,t} \\ u_{2,t} \\ u_{3,t} \\ u_{4,t} \end{bmatrix}}_{\mathbf{u}_t}, \tag{2.3}$$

where $\mathbf{A}$ is a transition matrix that depends on the time-discretization period T, $\mathbf{x}_{t-1}$ represents the position and velocity in the previous time step and $\mathbf{u}_t$ is a 4×1 real-valued Gaussian vector with zero mean and known covariance matrix, $\mathbf{\Sigma}_{\boldsymbol{u}}$. As a result, the noise term $\mathbf{Q}\mathbf{u}_t$ is a 4×1 real Gaussian vector, with zero mean and covariance matrix $\mathbf{Q}\mathbf{\Sigma}_u\mathbf{Q}^\top$, that represents the effect of unknown accelerations.

We assume that we receive RSS measurements coming from a WSN composed of J sensors. The measurement provided by the j-th sensor at time t is denoted as $y_{j,t}$ and the collection of J observations are represented as vector $\mathbf{y}_t = [y_{1,t}, y_{2,t}, y_{3,t} \dots y_{J,t}]^\top$. In order to describe the relationship between the observed RSS at sensor j, $y_{j,t}$, and the target position, $\mathbf{r}_t$, it is common to use the logarithmic path-loss model [97]

$$y_{j,t} = L_0 + \gamma_j 10 \log_{10}\left(\frac{d_0}{d_{j,t}}\right) + \varepsilon_{j,t}, \tag{2.4}$$

where $d_{j,t} = ||\mathbf{r}_t - \mathbf{s}_j||$ is the distance between the position of sensor j and the target at time t; d_0 is a known reference distance; L_0 is the path loss at the reference distance; γ_j is the path loss exponent and $\varepsilon_{j,t} \sim \mathcal{N}(\varepsilon_{j,t}; 0, \sigma^2_{\varepsilon_j})$ is normally distributed, zero-mean noise with variance $\sigma^2_{\varepsilon_j}$. Note that the noise contributions are assumed independent across different sensors and that the sensor positions, $\mathbf{s}_j, j = 1, 2, \dots, J$, are known.

Example 2: Coordinated turn model with angle of arrival (AoA) measurements.

The coordinated turn (CT) model is a mathematical representation that completely characterizes the dynamics of a target turning at an invariant

and known angle, ω [49]. The state equation then becomes

$$\underbrace{\begin{bmatrix} r_{1,t} \\ r_{2,t} \\ v_{1,t} \\ v_{2,t} \end{bmatrix}}_{\mathbf{x}_t} = \underbrace{\begin{bmatrix} 1 & 0 & \frac{\sin(\omega T)}{\omega} & -\frac{\cos(\omega T)-1}{\omega} \\ 0 & 1 & \frac{1-\cos(\omega T)}{\omega} & \frac{\sin(\omega T)}{\omega} \\ 0 & 0 & \cos(\omega T) & -\sin(\omega T) \\ 0 & 0 & \sin(\omega T) & \cos(\omega T) \end{bmatrix}}_{\mathbf{A}(\omega)} \underbrace{\begin{bmatrix} r_{1,t-1} \\ r_{2,t-1} \\ v_{1,t-1} \\ v_{2,t-1} \end{bmatrix}}_{\mathbf{x}_{t-1}} + \underbrace{\begin{bmatrix} \frac{1}{2}T^2 & 0 & 0 & 0 \\ 0 & \frac{1}{2}T^2 & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix} u_{1,t} \\ u_{2,t} \\ u_{3,t} \\ u_{4,t} \end{bmatrix}}_{\mathbf{u}_t}. \tag{2.5}$$

Compared to the CV model, the CT function has a transition matrix, $\mathbf{A}(\omega)$, that depends now on both the time-discretization period T and the turning angle, ω.

If we assume that we receive a bearing angle or AoA measurement coming from sensor j, the measurement equation becomes

$$y_{j,t} = \arctan\left(\frac{s_{2,j} - x_{2,t}}{s_{1,j} - x_{1,t}}\right) + \varepsilon_{j,t}, \quad j = 1, \ldots, J, \tag{2.6}$$

where $\varepsilon_{j,t} \sim \mathcal{N}(\varepsilon_{j,t}; 0, \sigma^2_{\varepsilon_j})$ is normally distributed, zero-mean noise with variance $\sigma^2_{\varepsilon_m}$ and $\mathbf{s}_j = [s_{1,j}, s_{2,j}]^\top$ is the position of the jth sensor in the two dimensional region.

Once we have a generic state-space model, we can define the Bayesian filtering problem where we seek filtered estimates of the state $\mathbf{x}_t$ based on the sequence of all available measurements up to time t, $\mathbf{y}_{1:t}$.

2.2.2 Bayesian filtering

Bayesian filtering is the process of recursively quantifying a degree of belief in the state $\mathbf{x}_t$ taking different values given the data, $\mathbf{y}_{1:t}$, up to time t. This process amounts to recursively constructing the posterior pdf $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ at each time instant t [10]. In order to do so the Bayesian filter assumes knowledge of three densities:

- $p(\mathbf{x}_0)$, the a priori target state density,
- $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, the state evolution or state transition density, and

- $p(\mathbf{y}_t|\mathbf{x}_t)$, the likelihood function.

Given the system model (target state and observations) in a state-space form, the transition density, $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, is characterized by the state equation (2.1) and the likelihood, $p(\mathbf{y}_t|\mathbf{x}_t)$, is described by the measurement equation (2.2). Note that, since the noise terms $\varepsilon_{j,t}$ are assumed to be independent across sensors, the observations $y_{1,t}, y_{2,t}, y_{3,t} \dots y_{J,t}$ become conditionally independent given the state $\mathbf{x}_t$, i.e., $p(\mathbf{y}_t|\mathbf{x}_t) = \prod_{j=1}^{J} p(y_{j,t}|\mathbf{x}_t)$.

The process of filtering consists in two steps performed in a recursive manner: prediction and update [99]. Assume that the posterior pdf at time instant $t-1$, $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$, is available. The prediction step involves using knowledge of the state equation to obtain the predictive density

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}. \tag{2.7}$$

At time instant t, when a new measurement, $\mathbf{y}_t$, becomes available, we proceed to update the filter with new information using Bayes' rule:

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{y}_{1:t}) &= \frac{p(\mathbf{y}_t|\mathbf{x}_t, \mathbf{y}_{1:t-1}) p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})} \\ &\propto p(\mathbf{y}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) \end{aligned} \tag{2.8}$$

where the (omitted) normalizing constant is

$$p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{y}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) d\mathbf{x}_t. \tag{2.9}$$

Once we have knowledge of the posterior density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, usually called the filtering density, we can compute optimal estimates of the state with respect to different criteria. For example, the minimum mean-square error (MMSE) estimate is the conditional mean of $\mathbf{x}_t$,

$$\hat{\mathbf{x}}_t^{MMSE} \triangleq \mathbb{E}\{\mathbf{x}_t|\mathbf{y}_{1:t}\} = \int \mathbf{x}_t p(\mathbf{x}_t|\mathbf{y}_{1:t}) d\mathbf{x}_t \tag{2.10}$$

where $\mathbb{E}\{\mathbf{x}_t|\mathbf{y}_{1:t}\}$ denotes the expected value of $\mathbf{x}_t$ conditioned upon $\mathbf{y}_{1:t}$. The maximum a posteriori (MAP) estimate, in the other hand, is the maximum of $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, i.e.,

$$\hat{\mathbf{x}}_t^{MAP} \triangleq \arg\max_{\mathbf{x}_t} \quad p(\mathbf{x}_t|\mathbf{y}_{1:t}). \tag{2.11}$$

The recurrence of (2.7) and (2.8) form the basis of the optimal Bayesian solution to the recursive filtering problem. An algorithm is said to be

optimal, in Bayesian filtering, when it obtains a complete characterization of $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ in a recursive manner. However this solution is simply conceptual in the sense that, in general, it cannot be determined analytically in closed form. Only in a restrictive set of cases, namely, when the state equation and the measurement equation are linear and Gaussian, or when the state space is discrete and finite, can the posterior be computed analytically. In the rest of the cases we have to resort to suboptimal techniques.

2.2.3 The Kalman filter and its extensions

The Kalman filter

When both the dynamic (2.1) and observation (2.2) models are linear and Gaussian, the densities $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, $t = 1, 2, \ldots$, are also Gaussian and can be exactly computed using the Kalman filter [99]. Specifically, in the Bayesian formulation of the Kalman filter we assume that

- $\mathbf{u}_t$ y $\boldsymbol{\varepsilon}_t$ are independent and have known Gaussian distributions,
- the dynamic model is a linear function of $\mathbf{x}_{t-1}$ and $\mathbf{u}_t$, and
- the observation model is a linear function of $\mathbf{x}_t$ and $\boldsymbol{\varepsilon}_t$.

Let us, thus, assume we have the linear state-space model

$$\begin{aligned} \mathbf{x}_t &= \mathbf{H}_t\mathbf{x}_{t-1} + \mathbf{W}_t\mathbf{u}_t, && (2.12)\\ \mathbf{y}_t &= \mathbf{G}_t\mathbf{x}_t + \mathbf{V}_t\boldsymbol{\varepsilon}_t, && (2.13) \end{aligned}$$

where the noise sequences are mutually independent, zero mean, white and Gaussian with identity covariance matrices, i.e., $\mathbf{u}_t \sim \mathcal{N}(0, \mathbf{I}_{n_x})$ and $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, \mathbf{I}_{n_y})$ and $\mathbf{H}_t$, $\mathbf{W}_t$, $\mathbf{G}_t$ and $\mathbf{V}_t$, are known matrices. Then, given $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{x}_{t-1}; \mathbf{x}_{t-1|t-1}; \mathbf{P}_{t-1|t-1})$, the recursive formulae of (2.7) and (2.8) yield

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) &= \mathcal{N}(\mathbf{x}_t; \mathbf{x}_{t|t-1}; \mathbf{P}_{t|t-1}), && (2.14)\\ p(\mathbf{x}_t|\mathbf{y}_{1:t}) &= \mathcal{N}(\mathbf{x}_t; \mathbf{x}_{t|t}; \mathbf{P}_{t|t}), && (2.15) \end{aligned}$$

where the prediction step (2.14) involves the computation of [3]

$$\begin{aligned} \hat{\mathbf{x}}_{t|t-1} &= \mathbf{H}_t\hat{\mathbf{x}}_{t-1|t-1}\\ \mathbf{P}_{t|t-1} &= \mathbf{H}_t\mathbf{P}_{t-1|t-1}\mathbf{H}_t^\top + \mathbf{W}_t\mathbf{W}_t^\top \end{aligned} \qquad (2.16)$$

and, for the update step, we compute

$$\begin{aligned}
\mathbf{S}_t &= \mathbf{G}_t \mathbf{P}_{t|t-1} \mathbf{G}_t^\top + \mathbf{V}_t \mathbf{V}_t^\top \\
\mathbf{K}_t &= \mathbf{P}_{t|t-1} \mathbf{G}_t^\top \mathbf{S}_t^{-1} \\
\hat{\mathbf{x}}_{t|t} &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t (\mathbf{y}_t - \mathbf{G}_t \hat{\mathbf{x}}_{t|t-1}) \\
\mathbf{P}_{t|t} &= \mathbf{P}_{t|t-1} - \mathbf{K}_t \mathbf{G}_t \mathbf{P}_{t|t-1}
\end{aligned} . \tag{2.17}$$

Matrix $\mathbf{K}_t$ is often termed the Kalman gain, while $\mathbf{S}_t$ is the covariance matrix of the innovation $\boldsymbol{\nu}_t = \mathbf{y}_t - \mathbf{G}_t \hat{\mathbf{x}}_{t|t-1}$. The recursive application of these equations gives us the mean and covariance matrix of the sequence of posterior probability distribution functions, namely $p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{x}_t; \hat{\mathbf{x}}_{t|t}, \mathbf{P}_{t|t})$, for $t = 1, 2, \ldots$.

Table 2.1 summarizes the Kalman filter algorithm for linear state-space models.

The extended Kalman filter

Many important real world applications are nonlinear and/or non-Gaussian, therefore the Kalman filter cannot be applied. A very popular approach to surmount this problem is the linearization of the state space model in order to later apply the Kalman filter to it. The main advantage of this so-called extended Kalman filter (EKF) [116] resides in the simplicity of the linear approximation. It consists in approximating $\mathbf{h}_t$ and $\mathbf{g}_t$ with a first-order Taylor series expansion and evaluating the linearized function at the current state estimate, either $\hat{\mathbf{x}}_{t-1|t-1}$ or $\hat{\mathbf{x}}_{t|t-1}$, respectively. To be specific, let us assume a general state-space model

$$\begin{aligned}
\mathbf{x}_t &= \mathbf{h}_t(\mathbf{x}_{t-1}) + \mathbf{W}_t \mathbf{u}_t, \\
\mathbf{y}_t &= \mathbf{g}_t(\mathbf{x}_t) + \mathbf{V}_t \boldsymbol{\varepsilon}_t,
\end{aligned} \tag{2.18}$$

where $\mathbf{h}_t$ and $\mathbf{g}_t$ are known differentiable nonlinear functions, the random sequences $\mathbf{u}_t$ and $\mathbf{v}_t$ are mutually independent zero-mean white Gaussian with identity covariance matrices, i.e., $\mathbf{u}_t \sim \mathcal{N}(0, \mathbf{I}_{n_x})$ and $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, \mathbf{I}_{n_y})$, and matrices $\mathbf{W}_t$ and $\mathbf{V}_t$ are also known. The EKF prediction step involves the computation of

$$\begin{aligned}
\hat{\mathbf{x}}_{t|t-1} &= \mathbf{h}_t(\hat{\mathbf{x}}_{t-1|t-1}) \\
\mathbf{P}_{t|t-1} &= \hat{\mathbf{H}}_t \mathbf{P}_{t-1|t-1} \hat{\mathbf{H}}_t^\top + \mathbf{W}_t \mathbf{W}_t^\top
\end{aligned} \tag{2.19}$$

Table 2.1: Kalman filter (KF)

At $t = 0$:

1. Assign a mean vector $\hat{\mathbf{x}}_0$ and a covariance matrix $\mathbf{P}_0$ to the first time instant $t = 0$, all taken from the prior distribution $p(\mathbf{x}_0; \hat{\mathbf{x}}_0, \mathbf{P}_0)$.

For $t > 0$:

1. Compute the predicted state vector, $\hat{\mathbf{x}}_{t|t-1}$, and predicted covariance matrix, $\hat{\mathbf{P}}_{t|t-1}$,

$$\begin{aligned}\hat{\mathbf{x}}_{t|t-1} &= \mathbf{H}_t \hat{\mathbf{x}}_{t-1|t-1}, \\ \mathbf{P}_{t|t-1} &= \mathbf{H}_t \mathbf{P}_{t-1|t-1} \mathbf{H}_t^\top + \mathbf{W}_t \mathbf{W}_t^\top.\end{aligned}$$

2. Collect a new observation $\mathbf{y}_t$.
3. Compute the covariance innovation matrix, $\mathbf{S}_t$, the Kalman gain, $\mathbf{K}_t$ and the posterior mean vector, $\hat{\mathbf{x}}_{t|t}$ and covariance matrix, $\mathbf{P}_{t|t}$,

$$\begin{aligned}\mathbf{S}_t &= \mathbf{G}_t \mathbf{P}_{t|t-1} \mathbf{G}_t^\top + \mathbf{V}_t \mathbf{V}_t^\top, \\ \mathbf{K}_t &= \mathbf{P}_{t|t-1} \mathbf{G}_t^\top \mathbf{S}_t^{-1}, \\ \hat{\mathbf{x}}_{t|t} &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t (\mathbf{y}_t - \mathbf{G}_t \hat{\mathbf{x}}_{t|t-1}), \\ \mathbf{P}_{t|t} &= \mathbf{P}_{t|t-1} - \mathbf{K}_t \mathbf{G}_t \mathbf{P}_{t|t-1}.\end{aligned}$$

and the update step is

$$\begin{aligned}
\mathbf{K}_t &= \mathbf{P}_{t|t-1}\hat{\mathbf{G}}_t^\top \mathbf{S}_t^{-1} \\
\mathbf{S}_t &= \hat{\mathbf{G}}_{t-1}\mathbf{P}_{t|t-1}\hat{\mathbf{G}}_{t-1}^\top + \mathbf{V}_t\mathbf{V}_t^\top \\
\hat{\mathbf{x}}_{t|t} &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{g}_t(\hat{\mathbf{x}}_{t|t-1})) \\
\mathbf{P}_{t|t} &= \mathbf{P}_{t|t-1} - \mathbf{K}_t\hat{\mathbf{G}}_t\mathbf{P}_{t|t-1}
\end{aligned} \tag{2.20}$$

where $\hat{\mathbf{H}}_t$ and $\hat{\mathbf{G}}_t$, are the Jacobian matrices of $\mathbf{h}_t$ and $\mathbf{g}_t$, respectively, evaluated at $\hat{\mathbf{x}}_{t-1|t-1}$ and $\hat{\mathbf{x}}_{t|t-1}$, i.e.,

$$\hat{\mathbf{H}}_t = \left[\begin{array}{ccc} \frac{\partial h_{1,t}(x_{1,t-1})}{\partial x_{1,t-1}} & \cdots & \frac{\partial h_{1,t}(x_{n_x,t-1})}{\partial x_{n_x,t-1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_{n_y,t}(x_{1,t-1})}{\partial x_{1,t-1}} & \cdots & \frac{\partial h_{n_y,t}(x_{n_x,t-1})}{\partial x_{n_x,t-1}} \end{array}\right]_{\mathbf{x}_{t-1}=\hat{\mathbf{x}}_{t-1|t-1}}$$

and

$$\hat{\mathbf{G}}_t = \left[\begin{array}{ccc} \frac{\partial g_{1,t}(x_{1,t})}{\partial x_{1,t}} & \cdots & \frac{\partial g_{1,t}(x_{n_x,t})}{\partial x_{n_x,t}} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_{n_y,t}(x_{1,t})}{\partial x_{1,t}} & \cdots & \frac{\partial g_{n_y,t}(x_{n_x,t})}{\partial x_{n_x,t}} \end{array}\right]_{\mathbf{x}_t=\hat{\mathbf{x}}_{t|t-1}}$$

Table 2.2 summarizes the Extended Kalman filter algorithm.

Even though the EKF produces a solution to the nonlinear problem that is easy to use, it has some important drawbacks. Indeed, the analytical linearization can produce highly unstable filters and for certain practical situations it may be difficult to compute the Jacobian matrices [114]. In particular, note that $\mathbf{h}_t$ and $\mathbf{g}_t$ must be differentiable to apply the approximation procedure. In general, the EKF tends to work well in systems with sufficiently slow dynamics.

The unscented Kalman filter

The unscented Kalman filter (UKF) was first introduced in [114, 66] in order to surmount the problems faced with the EKF and it has since become a very popular technique. The same as the EKF, it approximates the posterior density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ as a Gaussian. However it uses some deterministically chosen sample points, $\{\mathcal{X}^i\}_{i=0}^{N-1}$, and associated weights, $\{\mathcal{W}^i\}_{i=0}^{N-1}$, to characterize it. The main idea of the UKF is that, when propagated through a nonlinear function, these sample points capture the true mean and covariance. This is not necessarily true for the analytic approximations performed by the EKF.

Table 2.2: Extended Kalman filter (EKF)

At $t = 0$:

1. Assign a mean vector $\hat{\mathbf{x}}_0$ and a covariance matrix $\mathbf{P}_0$ to the first time instant $t = 0$, all taken from the prior distribution $p(\mathbf{x}_0; \hat{\mathbf{x}}_0, \mathbf{P}_0)$.

For $t > 0$:

1. Compute the Jacobian matrix of $\mathbf{h}_t$, denoted, $\hat{\mathbf{H}}_t$, and evaluate it at $\mathbf{x}_{t-1} = \hat{\mathbf{x}}_{t-1|t-1}$.
2. Compute the predicted state vector, $\hat{\mathbf{x}}_{t|t-1}$, and predicted covariance matrix, $\mathbf{P}_{t|t-1}$, as

$$\begin{aligned}\hat{\mathbf{x}}_{t|t-1} &= \mathbf{h}_t(\hat{\mathbf{x}}_{t-1|t-1}),\\ \mathbf{P}_{t|t-1} &= \hat{\mathbf{H}}_t \mathbf{P}_{t-1|t-1} \hat{\mathbf{H}}_t^\top + \mathbf{W}_t \mathbf{W}_t^\top.\end{aligned}$$

3. Compute the Jacobian matrix of $\mathbf{g}_t$, denoted, $\hat{\mathbf{G}}_t$, and evaluate it at $\mathbf{x}_t = \hat{\mathbf{x}}_{t|t-1}$.
4. Compute the covariance innovation matrix, $\mathbf{S}_t$, and the Kalman gain, $\mathbf{K}_t$, as

$$\begin{aligned}\mathbf{S}_t &= \hat{\mathbf{G}}_{t-1} \mathbf{P}_{t|t-1} \hat{\mathbf{G}}_{t-1}^\top + \mathbf{V}_t \mathbf{V}_t^\top,\\ \mathbf{K}_t &= \mathbf{P}_{t|t-1} \hat{\mathbf{G}}_t^\top \mathbf{S}_t^{-1}.\end{aligned}$$

5. Collect the new observation $\mathbf{y}_t$.
6. Compute the filter mean, $\hat{\mathbf{x}}_{t|t}$, and covariance matrix, $\mathbf{P}_{t|t}$,

$$\begin{aligned}\hat{\mathbf{x}}_{t|t} &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{g}_t(\hat{\mathbf{x}}_{t|t-1})),\\ \mathbf{P}_{t|t} &= \mathbf{P}_{t|t-1} - \mathbf{K}_t \hat{\mathbf{G}}_t \mathbf{P}_{t|t-1}.\end{aligned}$$

Specifically, given a set of N deterministic samples $\mathcal{X}_{t-1}^i$ and their weights $\mathcal{W}^i$, $i = 0, \ldots, N-1$, the prediction step involves the computation of

$$\begin{aligned} \hat{\mathbf{x}}_{t|t-1} &= \sum_{i=0}^{N-1} \mathcal{W}^i \mathbf{h}_t(\mathcal{X}_{t-1}^i), \qquad (2.21) \\ \mathbf{P}_{t|t-1} &= \mathbf{W}_t \mathbf{W}_t^\top + \\ &\quad \sum_{i=0}^{N-1} \mathcal{W}^i \left[\mathbf{h}_t(\mathcal{X}_{t-1}^i) - \hat{\mathbf{x}}_{t|t-1}\right] \left[\mathbf{h}_t(\mathcal{X}_{t-1}^i) - \hat{\mathbf{x}}_{t|t-1}\right].^\top \qquad (2.22) \end{aligned}$$

The sample points, $\mathcal{X}_{t-1}^i$, are then propagated through the nonlinear function, $\mathbf{h}_t$, in order to obtain the predictive *samples*

$$\mathcal{X}_{t|t-1}^i = \mathbf{h}_t(\mathcal{X}_{t-1}^i), \quad i = 1, \ldots, N, \qquad (2.23)$$

which, in turn, are used to compute the predictive observation

$$\hat{\mathbf{y}}_{t|t-1} = \sum_{i=0}^{N-1} \mathcal{W}^i \mathbf{g}_t(\mathcal{X}_{t|t-1}^i). \qquad (2.24)$$

Finally, the update step is

$$\begin{aligned} \hat{\mathbf{x}}_{t|t} &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}) \\ \mathbf{P}_{t|t} &= \mathbf{P}_{t|t-1} + \mathbf{K}_t \mathbf{S}_t \mathbf{K}_t^\top, \end{aligned} \qquad (2.25)$$

where the Kalman gain, $\mathbf{K}_t$, and the innovation covariance matrix, $\mathbf{S}_t$, are computed as

$$\begin{aligned} \mathbf{K}_t &= \mathbf{P}_{sz} \mathbf{S}_t^{-1} \\ \mathbf{S}_t &= \mathbf{V}_t \mathbf{V}_t^\top + \mathbf{P}_{zz} \end{aligned}$$

with

$$\mathbf{P}_{xz} = \sum_{i=0}^{N-1} \mathcal{W}^i \left(\mathcal{X}_{t|t-1}^i - \hat{\mathbf{x}}_{t|t-1}\right) \left(\mathbf{g}_t(\mathcal{X}_{t|t-1}^i) - \hat{\mathbf{y}}_{t|t-1}\right)^\top \qquad (2.26)$$

and

$$\mathbf{P}_{zz} = \sum_{i=0}^{N-1} \mathcal{W}^i \left(\mathbf{g}_t(\mathcal{X}_{t|t-1}^i) - \hat{\mathbf{y}}_{t|t-1}\right) \left(\mathbf{g}_t(\mathcal{X}_{t|t-1}^i) - \hat{\mathbf{y}}_{t|t-1}\right).^\top$$

Selection of sample points

The selection of sample points is based on the unscented transform [102] which is often used to represent probability distributions of random variables resulting from nonlinear transformations. Consider propagating a random variable $\mathbf{x}_t$ with mean $\hat{\mathbf{x}}_t$ and covariance $\mathbf{P}_t$ through an arbitrary nonlinear function $\mathbf{g}_t : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y}$ to produce a random variable $\mathbf{y}_t$, i.e.,

$$\mathbf{y}_t = \mathbf{g}_t(\mathbf{x}_t). \tag{2.27}$$

The first two moments of $\mathbf{y}_t$ can be computed as follows. First, $2n_x + 1$ weighted sample points, $(\mathcal{X}_t^i, \mathcal{W}_t^i)$, are chosen deterministically, so that they completely describe the true mean, $\hat{\mathbf{x}}_t$, and the true covariance, $\mathbf{P}_t$, of $\mathbf{x}_t$. A scheme to satisfy this requirement is [65]:

$$\begin{aligned}
\mathcal{X}_t^0 &= \hat{\mathbf{x}}_t & \mathcal{W}_t^0 &= \frac{\kappa}{(n_x+\kappa)} & i &= 0 \\
\mathcal{X}_t^i &= \hat{\mathbf{x}}_t + \left(\sqrt{(n_x+\kappa)\mathbf{P}_t}\right)_i & \mathcal{W}_t^i &= \frac{1}{2(n_x+\kappa)} & i &= 1,\ldots,n_x \\
\mathcal{X}_t^i &= \hat{\mathbf{x}}_t - \left(\sqrt{(n_x+\kappa)\mathbf{P}_t}\right)_i & \mathcal{W}_t^i &= \frac{1}{2(n_x+\kappa)} & i &= n_x+1,\ldots,2n_x
\end{aligned}$$

where κ is a scaling parameter such that $\kappa + n_x \neq 0$ and $\left(\sqrt{(n_x+\kappa)\mathbf{P}_t}\right)_i$ is the ith row of the matrix $\mathbf{L}$, that satisfies $(n_x+\kappa)\mathbf{P}_t = \mathbf{L}^\top\mathbf{L}$ (the square root of $(n_x+\kappa)\mathbf{P}_t$). The weights are normalized to yield $\sum_{i=0}^{2n_x} \mathcal{W}_t^i = 1$. Next, each point is propagated through the nonlinear function $\mathbf{g}_t$ to obtain

$$\mathcal{Y}_t^i = \mathbf{g}_t(\mathcal{X}_t^i), \quad i = 1,\ldots,2n_x+1. \tag{2.28}$$

The first two moments of $\mathbf{y}_t$ are then computed as

$$\begin{aligned}
\hat{\mathbf{y}}_t &= \sum_{i=0}^{2n_x} \mathcal{W}_t^i \mathcal{Y}_t^i, \\
\mathbf{P}_t^y &= \sum_{i=0}^{2n_x} \mathcal{W}_t^i \left[\mathcal{Y}^i - \hat{\mathbf{y}}_t\right]\left[\mathcal{Y}^i - \hat{\mathbf{y}}_t\right].^\top
\end{aligned} \tag{2.29}$$

The UKF algorithm has been broadly used in various applications of nonlinear filtering [65] because it often provides an attractive trade-off between accuracy and computational complexity. Improved UKFs have also been proposed that introduce a scaling factor in the weights (known as the scaled UKF) or where the state is redefined as the concatenation

of the original state and the noise variables in order to better capture the uncertainty [114]. Note, however, that the same as other variants of the original KF, the UKF relies on a Gaussian approximation of the true posterior distribution, hence it is not suitable for problems where the latter distribution can be multimodal.

Table 2.3 outlines the UKF algorithm.

2.3 Nonlinear filtering via sequential importance sampling

In this section we consider the use of Monte Carlo (i.e. simulation-based) methods for nonlinear filtering. When it is possible to draw exactly from the distribution of interest, it is straight forward to approximate any complicated integrals as empirical averages using i.i.d. samples. However, in the context of nonlinear filtering, it is generally impossible to draw exactly from the posterior distribution of interest. In order to surmount this problem we introduce the importance sampling methodology [101]. This scheme solves the problem of sampling from complicated distributions, however it is computationally prohibitive when applied to the filtering problem [41]. Hence, we move on to explain the sequential importance sampling methodology that approximates the posterior distribution in a fast recursive manner. Finally, we devote a section to describe some state-of-the-art sequential Monte Carlo algorithms, also known as particle filters, used for nonlinear tracking [22].

2.3.1 Exact Monte Carlo sampling

Assume we are interested in approximating an integral of the form $\int f(\mathbf{x})p(\mathbf{x})d\mathbf{x}$ where f is an integrable function and $p(\mathbf{x})$ is a pdf of interest. If one can simulate M samples distributed exactly according to $p(\mathbf{x})$, then it is straightforward to approximate

$$I(f) = \int f(\mathbf{x})p(\mathbf{x})d\mathbf{x} \approx \frac{1}{M}\sum_{i=1}^{M} f(\mathbf{x}^{(i)}) \tag{2.30}$$

and under mild assumptions [32], the sum converges to the integral as $M \to \infty$.

In the Bayesian target tracking problem, the aim is to estimate recursively in time the distribution with pdf $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, the marginal with

Table 2.3: Unscented Kalman filter (UKF)

At $t = 0$:

1. Assign a mean vector $\hat{\mathbf{x}}_0$ and a covariance matrix $\mathbf{P}_0$ to the first time instant $t = 0$, all taken from the prior distribution $p(\mathbf{x}_0; \hat{\mathbf{x}}_0, \mathbf{P}_0)$.

For $t > 0$:

1. Compute the predicted state vector, $\hat{\mathbf{x}}_{t|t-1}$, and predicted covariance matrix, $\mathbf{P}_{t|t-1}$,

$$\begin{aligned}
&\hat{\mathbf{x}}_{t|t-1} = \sum_{i=0}^{N-1} \mathcal{W}_{t-1}^i \mathbf{h}_t(\mathcal{X}_{t-1}^i), \\
&\mathbf{P}_{t|t-1} = \mathbf{W}_t \mathbf{W}_t^\top \\
&+ \sum_{i=0}^{N-1} \mathcal{W}_{t-1}^i \left[\mathbf{h}_t(\mathcal{X}_{t-1}^i) - \hat{\mathbf{x}}_{t|t-1}\right] \left[\mathbf{h}_t(\mathcal{X}_{t-1}^i) - \hat{\mathbf{x}}_{t|t-1}\right]^\top.
\end{aligned}$$

2. Compute predicted samples,

$$\mathcal{X}_{t|t-1}^i = \mathbf{h}_t(\mathcal{X}_{t-1}^i).$$

3. Compute predicted measurement,

$$\hat{\mathbf{y}}_{t|t-1} = \sum_{i=0}^{N-1} \mathcal{W}_{t-1}^i \mathbf{g}_t(\mathcal{X}_{t|t-1}^i).$$

4. Compute the covariance innovation matrix, $\mathbf{S}_t$, and Kalman gain, $\mathbf{K}_t$,

$$\begin{aligned}
\mathbf{P}_{xz} &= \sum_{i=0}^{N-1} \mathcal{W}_{t-1}^i \left(\mathcal{X}_{t|t-1}^i - \hat{\mathbf{x}}_{t|t-1}\right) \left(\mathbf{g}_t(\mathcal{X}_{t|t-1}^i) - \hat{\mathbf{y}}_{t|t-1}\right)^\top, \\
\mathbf{P}_{zz} &= \sum_{i=0}^{N-1} \mathcal{W}_{t-1}^i \left(\mathbf{g}_t(\mathcal{X}_{t|t-1}^i) - \hat{\mathbf{y}}_{t|t-1}\right) \left(\mathbf{g}_t(\mathcal{X}_{t|t-1}^i) - \hat{\mathbf{y}}_{t|t-1}\right)^\top, \\
\mathbf{S}_t &= \mathbf{V}_t \mathbf{V}_t^\top + \mathbf{P}_{zz}, \\
\mathbf{K}_t &= \mathbf{P}_{sz} \mathbf{S}_t^{-1}.
\end{aligned}$$

5. Collect the new observation $\mathbf{y}_t$.
6. Compute the filter mean, $\hat{\mathbf{x}}_{t|t}$, and covariance matrix, $\mathbf{P}_{t|t}$,

$$\begin{aligned}
\hat{\mathbf{x}}_{t|t} &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}), \\
\mathbf{P}_{t|t} &= \mathbf{P}_{t|t-1} + \mathbf{K}_t \mathbf{S}_t \mathbf{K}_t^\top.
\end{aligned}$$

density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ and expectations of the form

$$I_{0:t}(f) = \mathbb{E}_{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}(f(\mathbf{x}_{0:t})) = \int f(\mathbf{x}_{0:t})p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t}, \tag{2.31}$$

where $f : \mathbb{R}^{(t+1)n_x} \rightarrow \mathbb{R}$ is a generic integrable function. In order to apply exact Monte Carlo sampling to this problem, we should be able to simulate M i.i.d random samples $\{\mathbf{x}_{0:t}^{(i)}; i = 1, \dots M\}$ according to the density $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. Then, an empirical estimate of the measure $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t}$ is given by

$$p^M(d\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{1}{M}\sum_{i=1}^{M} \delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t}), \tag{2.32}$$

where $\delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t})$ is a Dirac delta measure located at $\mathbf{x}_{0:t}^{(i)}$. Given the approximation of the joint distribution, one can approximate the integral of (2.31) as

$$I_{0:t}^M(f) = \int f(\mathbf{x}_{0:t})p^M(d\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{1}{M}\sum_{i=1}^{M} f(\mathbf{x}_{0:t}^{(i)}). \tag{2.33}$$

It can be proven using the strong law of large numbers [32] that

$$\lim_{M\rightarrow\infty} I_{0:t}^M(f) = I_{0:t}(f) \tag{2.34}$$

almost surely (a.s.) under mild conditions [32]. Unfortunately, it is impossible to draw exactly from $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ in most practical cases (and inefficient for large t in *all* cases).

2.3.2 Importance sampling

An alternative approach to the direct sampling from $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ is the use of the importance sampling (IS) methodology [101]. The core idea of IS lies in using a sampling function different from the one of interest and then measuring "how well" the samples describe the true distribution. The chosen function must fulfill certain simple conditions and it is selected so that one can easily draw from it.

Specifically, we choose a so-called importance function, $\pi(\mathbf{x})$, that has the same support as the density $p(\mathbf{x}_t)$, i.e., $\pi(\mathbf{x}) > 0$ whenever $p(\mathbf{x}) > 0$. This function is inserted in the integral of (2.30) as

$$I(f) = \int f(\mathbf{x})p(\mathbf{x})d\mathbf{x} = \frac{\int f(\mathbf{x})p(\mathbf{x})d\mathbf{x}}{\int p(\mathbf{x})d\mathbf{x}} = \frac{\int f(\mathbf{x})c\frac{p(\mathbf{x})}{\pi(\mathbf{x})}\pi(\mathbf{x})d\mathbf{x}}{\int c\frac{p(\mathbf{x})}{\pi(\mathbf{x})}\pi(\mathbf{x})d\mathbf{x}} \tag{2.35}$$

where c is an arbitrary constant. The advantage of this rearrangement is that it is enough to be able to evaluate $p(\mathbf{x})$ and $\pi(\mathbf{x})$ up to an arbitrary (and even unknown) proportionality constant c, as in the expression of (2.35) the constants cancel out. If we simulate M samples distributed according to $\pi(\mathbf{x})$, then we can approximate the integral of (2.35) as

$$I(f) \approx \frac{\frac{1}{M}\sum_{i=1}^{M} f(\mathbf{x}^{(i)})w^{(i)*}}{\frac{1}{M}\sum_{j=1}^{M} w^{(i)*}} = \sum_{i=1}^{M} f(\mathbf{x}^{(i)})\frac{w^{(i)*}}{\sum_{j=1}^{M} w^{(j)*}}, \tag{2.36}$$

where

$$w^{(i)*} = c\frac{p(\mathbf{x}^{(i)})}{\pi(\mathbf{x}^{(i)})} \tag{2.37}$$

is the unnormalized importance weight of the sample $\mathbf{x}^{(i)}$.

Applied to the Bayesian tracking problem, we draw M samples from the importance function $\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ and obtain an approximation of (2.33) as

$$I_{0:t}^{M}(f) = \sum_{i=1}^{M} f(\mathbf{x}_{0:t}^{(i)})\frac{w_t^{(i)*}}{\sum_{j=1}^{M} w_t^{(j)*}} = \sum_{i=1}^{M} f(\mathbf{x}_{0:t}^{(i)})w_t^{(i)} \tag{2.38}$$

where the unnormalized importance weights can be computed as

$$\begin{aligned} w_t^{(i)*} &= \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t}^{(i)})p(\mathbf{x}_{0:t}^{(i)})}{\pi(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}, \\ &\propto \frac{p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}{\pi(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})} \end{aligned} \tag{2.39}$$

and $\{w_t^{(i)}, i = 1, \ldots, M\}$ are normalized importance weights, i.e.,

$$w_t^{(i)} = \frac{w_t^{(i)*}}{\sum_{j=1}^{M} w_t^{(j)*}}. \tag{2.40}$$

The IS method provides a solution to the problem of sampling from functions that do not have a closed form, however, it has a main drawback for filtering applications. In order to compute the weights at time t, one has to take into account the whole sequence of sample trajectories $\{\mathbf{x}_{0:t}^{(i)}\}$ when we are truly interested in a sequential and recursive solution. In order to avoid this problem there is a sequential version of the IS method that fits nicely in the Bayesian filtering framework.

2.3.3 Sequential importance sampling: particle filtering

The sequential importance sampling (SIS) algorithm, is a recursive version of the IS technique [41]. If we choose an importance function that can be factorized as

$$\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \pi(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})\pi(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{1:t}) \tag{2.41}$$

where $\pi(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{0:t})$ is the importance function for $\mathbf{x}_t$ conditioned upon $\mathbf{x}_{0:t-1}$ and $\mathbf{y}_{1:t}$, then we can compute the importance function from time 0 to time t in a recursive manner.

If the factorization (2.41) holds, the weights can also be computed recursively. If we expand (2.39) using Bayes' rule and factorize the importance function we obtain

$$\begin{aligned} w_t^{(i)*} &\propto \frac{p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}{\pi(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}, \\ &\propto \frac{p(\mathbf{y}_t|\mathbf{x}_{0:t}^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)},\mathbf{y}_{1:t-1})}{\pi(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)},\mathbf{y}_{1:t})} \frac{p(\mathbf{x}_{0:t-1}^{(i)}|\mathbf{y}_{1:t-1})}{\pi(\mathbf{x}_{0:t-1}^{(i)}|\mathbf{y}_{1:t-1})} \\ &\propto \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{\pi(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)},\mathbf{y}_{1:t})} w_{t-1}^{(i)}, \end{aligned} \tag{2.42}$$

where in order to reduce $p(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)},\mathbf{y}_{1:t-1})$ to $p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})$ we have taken into account the fact that the dynamic model specified by the state space model in (2.1) is Markov. Note that this new equation to evaluate the weights is recursive in the sense that it updates the weights with the new information and does not require to reprocess data from time $t = 0$.

Table 2.4 describes the basic SIS algorithm. In this context, the pair of sample and weight, $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$, is often termed "particle" and the SIS algorithm a "particle filter" (PF). The numerical complexity of this algorithm is $\mathcal{O}(\mathcal{M})$ [39], however note that it is completely parallelizable. The estimation process is not included in the algorithm description as it is not required for the algorithm to run, however at any time that an estimation of $I(f)$ is required it would be obtained as

$$I_{0:t}^M(f) = \sum_{i=1}^{M} f(\mathbf{x}_{0:t}^{(i)}) w_t^{(i)}. \tag{2.43}$$

Table 2.4: Sequential importance sampling (SIS)

At $t = 0$, for $i = 1, \ldots, M$:

1. Sample $x_0^{(i)} \sim \pi(\mathbf{x}_0)$.
2. Set $w_0^{(i)} = \frac{1}{M}$.

For $t > 0$ and $i = 1, \ldots, M$:

1. Sample $\mathbf{x}_t^{(i)} \sim \pi(\mathbf{x}_t | \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})$ and set $\mathbf{x}_{0:t}^{(i)} \triangleq (\mathbf{x}_{0:t-1}^{(i)}, \mathbf{x}_t^{(i)})$.
2. Evaluate the weights up to a normalizing constant,

$$w_t^{(i)*} = w_{t-1}^{(i)} \frac{p(\mathbf{y}_t | \mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)} | \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t-1})}{\pi(\mathbf{x}_t^{(i)} | \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})}.$$

3. Normalize the importance weights,

$$w_t^{(i)} = \frac{w_t^{(i)*}}{\sum_{j=1}^{M} w_t^{(j)*}}.$$

Degeneracy of the algorithm

It is well known that the sequential application of the importance sampling methodology with a finite number of samples, $M < \infty$, quickly leads to a degenerate set of particles [41]. This problem is know as the degeneracy problem of the SIS methodology and in practice translates to all but one normalized weights having a value close to zero. This difficulty is commonly overcome by adding a resampling step. Resampling is the process of randomly discarding particle trajectories with low weights and replicating particles with high weights [99]. Intuitively, it produces i.i.d samples from the approximate discrete measure $p^M(d\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ given by the particles. A resampling step can be taken every time the approximate effective sample size [39]

$$\hat{M}_{eff} = \frac{1}{\sum_{i=1}^{M}(w_t^{(i)})^2} \tag{2.44}$$

falls below a user defined threshold. Since $\hat{M}_{eff} \leq M$, typical threshold values are λM for some $0 < \lambda < 1$. The resulting algorithm is commonly referred to as sequential importance resampling (SIR) or sequential importance sampling with resampling (SIS/R).

In order to perform resampling it is necessary to process the weights of all particles jointly in order to select the surviving particles. Unfortunately, this means that the SIS with resampling is no longer paralellizable [45]. Recently, however, some novel resampling schemes that can be implemented in a distributed manner have been suggested [14]. This will be the core of our ideas for the distributed particle filters we propose in Chapters 4 and 5.

Selection of the importance function

The choice of importance function affects greatly the performance of the algorithm. In fact, it characterizes so much the algorithm that some particle filters have a specific name just due to the choice of importance function, e.g., the auxiliary particle filter, the unscented particle filter, the gaussian particle filter, etc.

The simplest choice of importance function is the prior distribution of the state-space model. In this case we have $\pi(\mathbf{x}_t|\mathbf{x}_{0:t}, \mathbf{y}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$. Even though this choice is inefficient as the state space is explored without taking into account any knowledge of the observations, it leads to a simple expression for the weights

$$w_t^{(i)*} \propto w_{t-1}^{(i)} p(\mathbf{y}_t|\mathbf{x}_t^{(i)}), \tag{2.45}$$

and thus is quite appealing for many practical applications.

If resampling is carried out at every time step, i.e., without regard of the effective sample size, then the resulting procedure coincides with the *bootstrap filter*, originally proposed in [47].

Table 2.5, describes an SIR algorithm whose selected importance function is the prior. Note that the bar in $\bar{\mathbf{x}}_t^{(i)}$ and $\bar{w}_t^{(i)}$ indicates that the particles have not yet gone through the resampling step.

2.4 Improved particle filters

2.4.1 SIS with optimal importance function

The optimal importance function is defined in [41] as the density function that minimizes the variance of the importance weights conditioned upon $\mathbf{x}_{t-1}^{(i)}$ and $\mathbf{y}_{1:t}$. For a Markov model with conditional independent observations, it takes the form

$$\pi(\mathbf{x}_t|\mathbf{x}_{0:t}, \mathbf{y}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t). \tag{2.46}$$

With this optimal proposal pdf, the computation of the weights reduces to

$$w_t^{(i)*} \propto w_{t-1}^{(i)} \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)} = w_{t-1}^{(i)} p(\mathbf{y}_t|\mathbf{x}_{t-1}^{(i)}) \tag{2.47}$$

where the equality is obtained from the relationship

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{y}_t|\mathbf{x}_{t-1})}.$$

A distinctive feature of this approach is that the weight $w_t^{(i)*}$ does *not* depend on $\mathbf{x}_t^{(i)}$, but only on $\mathbf{x}_{t-1}^{(i)}$.

The optimal importance function suffers from two major drawbacks, though. It requires the ability to sample from $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t)$ and to evaluate (up to a proportionality constant) the integral

$$p(\mathbf{y}_t|\mathbf{x}_{t-1}^{(i)}) = \int p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})d\mathbf{x}_t$$

which, in general, does not admit a closed form. An analytical evaluation is only possible for the special class of partial linear and Gaussian state space models, in which the state equation is Gaussian and the observation

Table 2.5: Sequential importance resampling (SIR) with a prior importance function.

At $t = 0$, for $i = 1, \ldots, M$:

1. Sample $x_0^{(i)} \sim \pi(\mathbf{x}_0)$.
2. Set $w_0^{(i)} = \frac{1}{M}$.

For $t > 0$, for $i = 1, \ldots, M$:

1. For $i = 1, \ldots, M$, sample $\bar{\mathbf{x}}_t^{(i)} \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(i)})$ and set $\bar{\mathbf{x}}_{0:t}^{(i)} \triangleq (\mathbf{x}_{0:t-1}^{(i)}, \bar{\mathbf{x}}_t^{(i)})$.
2. For $i = 1, \ldots, M$, evaluate the weights up to a normalizing constant,
$$\bar{w}_t^{(i)*} = w_{t-1}^{(i)} p(\mathbf{y}_t | \bar{\mathbf{x}}_t^{(i)}).$$
3. For $i = 1, \ldots, M$, normalize the importance weights,
$$\bar{w}_t^{(i)} = \frac{\bar{w}_t^{(i)*}}{\sum_{j=1}^{M} \bar{w}_t^{(j)*}}.$$
4. Compute $\widehat{M}_{eff}$.
5. If $\widehat{M}_{eff} \leq M_{thres}$ resample:
   - Draw indices $k_1, ..., k_M \in \{1, ..., M\}$ according to the probabilities $\bar{w}_t^{(1)}, ..., \bar{w}_t^{(M)}$.
   - Set $\left(\mathbf{x}_t^{(i)}\right) = \left(\bar{\mathbf{x}}_t^{(k_i)}\right)$ with probability $\bar{w}_t^{(k_i)}$, for $i = 1, ..., M$.
   - Set $w_t^{(i)} = 1/M$, for i=1, ..., M.
6. Otherwise set $\left(\mathbf{x}_t^{(i)}\right) = \left(\bar{\mathbf{x}}_t^{(i)}\right)$, for i=1, ..., M. .

equation is both linear and Gaussian. In such cases, $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t)$ is also Gaussian and can be computed analytically. When this is not the case, an alternative approach is to approximate the functions $\mathbf{h}_t$ and $\mathbf{g}_t$ by way of a first order Taylor expansion around a suitable linearization point $\hat{\mathbf{x}}_{t|t-1}$, in order to approximate the optimal importance function.

2.4.2 The auxiliary particle filter

The auxiliary particle filter (APF) was originally proposed in [94] as an extension of the bootstrap filter of [47]. The key element of the APF is an importance function that generates samples in a state space extended with one extra dimension. This dimension corresponds to a random index that identifies the particles at time $t-1$ that best "matches" the observations at time t.

The bootstrap filter, though, is derived to approximate $p(\mathbf{x}_t|\mathbf{y}_{1:t})d\mathbf{x}_t$. We, on the other hand, work with the more general sequential importance sampling/resampling (SIS/R) framework, which requires the APF to be introduced in a different way. In particular, we propose an APF where the use of auxiliary random indices is restricted to the resampling step. This is not the only way in which an auxiliary SMC algorithm can be constructed for our problem [94]. However, the proposed formulation is simple and does not significantly increase the computational complexity.

To be specific, let $\{\bar{\mathbf{x}}_{0:t}^{(i)}\}_{i=1}^{M}$ be the particles available at time t before resampling, with importance weights denoted $\bar{w}_t^{(i)}$, $i = 1, ..., M$. We propose to take into account the observation vector $\mathbf{y}_{t+1}$ in the resampling step at time t. This is done by constructing normalized auxiliary weights of the form

$$\lambda_t^{(i)} \propto \bar{w}_t^{(i)} p(\mathbf{y}_{t+1}|\overline{\mathbf{x}}_{t+1|t}^{(i)}), \tag{2.48}$$

for $i = 1, ..., M$, where $\overline{\mathbf{x}}_{t+1|t}^{(i)}$ is some characterization of $\mathbf{x}_{t+1}^{(i)}$ given $\bar{\mathbf{x}}_t^{(i)}$. This characterization could be a sample from the prior $\overline{\mathbf{x}}_{t+1|t}^{(i)} \sim p(\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t^{(i)})$ or the predictive mean, $\overline{\mathbf{x}}_{t+1|t}^{(i)} = \mathbb{E}[\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t^{(i)}]$. Then, we perform importance resampling according to the auxiliary weights, i.e., we draw

$$\mathbf{x}_{0:t}^{(i)} = \bar{\mathbf{x}}_{0:t}^{(k)} \tag{2.49}$$

with probability $\lambda_t^{(k)}$, for $i = 1, ..., M$ and $k \in \{1, ..., M\}$ (equivalently, we generate M samples from the discrete distribution given by the probabilities $P\left\{\bar{\mathbf{x}}_{0:t}^{(k)}, \right\} = \lambda_t^{(k)}$, $k = 1, ..., M$). The resulting normalized importance

weights, after resampling, are

$$w_t^{(i)} \propto \frac{\bar{w}_t^{(i)}}{\lambda_t^{(i)}} = \frac{1}{p(\mathbf{y}_{t+1}|\overline{\mathbf{x}}_{t+1|t}^{(k_i)})}, \tag{2.50}$$

where $k_i \in \{1, ..., M\}$ is the index such that $\mathbf{x}_{0:t}^{(i)} = \bar{\mathbf{x}}_{0:t}^{(k_i)}$. The weights, $w_t^{(i)}$, and the resampled particles, $\mathbf{x}_{0:t}^{(i)}$, $i = 1, ..., M$, are carried over to the next iteration of the algorithm.

Table 2.6 describes the resulting algorithm.

2.4.3 The Rao-Blackwellized particle filter

In some state space models, it is possible to integrate out a subset of the state variables using a (conditionally) optimal filter. This technique is known as Rao-Blackwellization [25]. Assume that the state vector, $\mathbf{x}_t$, can be divided into two subvectors, $\mathbf{x}_{1,t}$ and $\mathbf{x}_{2,t}$, such that $\mathbf{x}_t = [\mathbf{x}_{1,t}^\top, \mathbf{x}_{2,t}^\top]^\top$, and assume also that we can express the two sequences corresponding to the subvectors separately, that is, $\mathbf{x}_{j,0:t} = (\mathbf{x}_{j,0}, \dots \mathbf{x}_{j,t})$, for $j = 1, 2$. In this case the estimation of the integral of (2.31) can be rewritten as follows

$$\begin{aligned}
I_{0:t}(f) &= \int f(\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t} \\
&= \int f(\mathbf{x}_{0:t}) p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}) d\mathbf{x}_{0:t} \\
&= \int \Big[\int f(\mathbf{x}_{1,0:t}, \mathbf{x}_{2,0:t}) p(\mathbf{y}_{1:t}|\mathbf{x}_{1,0:t}, \mathbf{x}_{2,0:t}) \\
&\qquad \times p(\mathbf{x}_{2,0:t}|\mathbf{x}_{1,0:t}) d\mathbf{x}_{2,0:t} \Big] p(\mathbf{x}_{1,0:t}) d\mathbf{x}_{1,0:t} \\
&= \int g(\mathbf{x}_{1,0:t}) p(\mathbf{x}_{1,0:t}) d\mathbf{x}_{1,0:t}
\end{aligned} \tag{2.51}$$

where

$$g(\mathbf{x}_{1,0:t}) \triangleq \int f(\mathbf{x}_{1,0:t}, \mathbf{x}_{2,0:t}) p(\mathbf{y}_{1:t}|\mathbf{x}_{1,0:t}, \mathbf{x}_{2,0:t}) p(\mathbf{x}_{2,0:t}|\mathbf{x}_{1,0:t}) d\mathbf{x}_{2,0:t}. \tag{2.52}$$

If we can solve the integral in (2.52) exactly, then $\mathbf{x}_{2,0:t}$ is marginalized out. The SMC algorithms that rely on this technique are commonly known as Rao-Blackwellized particle filtering (RBPF) algorithms.

Table 2.6: Auxiliary particle filter for SIR (A-SIR)

At $t = 0$, for $i = 1, \ldots, M$:

1. Sample $x_0^{(i)} \sim \pi(\mathbf{x}_0)$.
2. Set $w_0^{(i)} = \frac{1}{M}$.

For $t > 0$, and $i = 1, \ldots, M$:

1. Draw $\bar{\mathbf{x}}_t^{(i)}$ from the pdf $p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$.
2. Update the weights, $\bar{w}_t^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{y}_t|\bar{\mathbf{x}}_t^{(i)})$
3. Compute predictions $\overline{\mathbf{x}}_{t+1|t}^{(i)}$, e.g. $\overline{\mathbf{x}}_{t+1|t}^{(i)} = \mathbb{E}[\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t^{(i)}]$.
4. Resample always performing the following steps:
   - Compute the unnormalized auxiliary weights $\lambda_t^{(i)*} = \bar{w}_t^{(i)} p(\mathbf{y}_{t+1}|\overline{\mathbf{x}}_{t+1|t}^{(i)})$.
   - Normalize the auxiliary weights,

$$\lambda_t^{(i)} = \frac{\lambda_t^{(i)*}}{\sum_{j=1}^{M} \lambda_t^{(j)*}}.$$

   - Draw indices $k_1, ..., k_M \in \{1, ..., M\}$ according to the probabilities $\lambda_t^{(1)}, ..., \lambda_t^{(M)}$.
   - Set $\mathbf{x}_{0:t}^{(i)} = \bar{\mathbf{x}}_{0:t}^{(k_i)}$ with probability $\lambda_t^{(k_i)}$, for $i = 1, ..., M$.
   - Update the weights as $w_t^{(i)} = \frac{1}{p(\mathbf{y}_{t+1}|\overline{\mathbf{x}}_{t+1|t}^{(k_i)})}$, for $i = 1, ..., M$.

Since it is handled analytically, a RBPF algorithm no longer needs to sample the marginalized vector, $\mathbf{x}_{2,t}$. Therefore the posterior distribution that the particle filter aims at approximating is

$$p(\mathbf{x}_{1,0:t}|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_{1,0:t}, \mathbf{y}_{1:t-1})p(\mathbf{x}_{1,t}|\mathbf{x}_{1,0:t-1}, \mathbf{y}_{1:t-1})p(\mathbf{x}_{1,0:t-1}|\mathbf{y}_{1:t-1}), \quad (2.53)$$

where the likelihood term, $p(\mathbf{y}_t|\mathbf{x}_{1,0:t}, \mathbf{y}_{1:t-1})$, and the prior density, $p(\mathbf{x}_{1,t}|\mathbf{x}_{1,0:t-1}, \mathbf{y}_{1:t-1})$, are integrals with respect to the conditional posterior of $\mathbf{x}_{2,0:t}$, namely

$$p(\mathbf{y}_t|\mathbf{x}_{1,0:t}, \mathbf{y}_{1:t-1}) = \int p(\mathbf{y}_t, \mathbf{x}_{2,t}|\mathbf{x}_{1,0:t}, \mathbf{y}_{1:t-1})d\mathbf{x}_{2,0:t}, \quad (2.54)$$

and

$$p(\mathbf{x}_{1,t}|\mathbf{x}_{1,0:t-1}, \mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_{1,t}, \mathbf{x}_{2,t-1}, |\mathbf{x}_{1,0:t-1}, \mathbf{y}_{1:t-1})d\mathbf{x}_{2,t-1}. \quad (2.55)$$

Note that due to the fact that the likelihood and the prior are integrals over $\mathbf{x}_{2,t}$ and $\mathbf{x}_{2,t-1}$, now $\mathbf{x}_{1,t}$ is not Markov and the observations $\mathbf{y}_{t-1}$ are not conditionally independent anymore.

Given that we want to approximate (2.53) with a particle filter, the weights of a RBPF algorithm can be computed as

$$w_t^{(i)*} \propto \frac{p(\mathbf{y}_t|\mathbf{x}_{0:t}^{(i)}, \mathbf{y}_{1:t-1})p(\mathbf{x}_{1,t}^{(i)}|\mathbf{x}_{1,0:t-1}^{(i)}, \mathbf{y}_{1:t-1})}{\pi(\mathbf{x}_{1,t}^{(i)}|\mathbf{x}_{1,0:t-1}^{(i)}, \mathbf{y}_{1:t-1})} w_{t-1}^{(i)} \quad (2.56)$$

where the computation of $p(\mathbf{x}_{1,t}^{(i)}|\mathbf{x}_{1,0:t-1}^{(i)}, \mathbf{y}_{1:t-1})$ and $p(\mathbf{y}_t|\mathbf{x}_{0:t}^{(i)}, \mathbf{y}_{1:t-1})$ depends on the analytic solution obtained for $p(\mathbf{x}_{2,t}|\mathbf{x}_{1,0:t}^{(i)}, \mathbf{y}_{1:t})$ and $\pi(\mathbf{x}_{1,t}^{(i)}|\mathbf{x}_{1,0:t-1}^{(i)}, \mathbf{y}_{1:t-1})$ is the proposal function.

Table 2.7 describes the general steps of a RBPF algorithm. Note that step 2 should be recursive as well if the method is to be practical. This normally implies that some statistics characterizing $p(\mathbf{x}_{2,t}|\mathbf{x}_{1,0:t}^{(i)}, \mathbf{y}_{1:t})$ have to be stored.

One example of state space model that allows this type of marginalization is the conditionally linear Gaussian state space model [41]. Let $n_{1,x}$ be the dimension of $\mathbf{x}_{1,t}$ and consider the model

$$\begin{aligned} \mathbf{x}_{1,t} &= \mathbf{f}(\mathbf{x}_{1,t-1}, \mathbf{v}_t) \\ \mathbf{x}_{2,t} &= \mathbf{H}_t(\mathbf{x}_{1,t})\mathbf{x}_{2,t-1} + \mathbf{W}_t(\mathbf{x}_{1,t})\mathbf{u}_t, \\ \mathbf{y}_t &= \mathbf{G}_t(\mathbf{x}_{1,t})\mathbf{x}_{2,t} + \mathbf{V}_t\boldsymbol{\varepsilon}_t, \end{aligned} \quad (2.57)$$

where $\mathbf{f}$ is a possibly nonlinear function of the state variable $\mathbf{x}_{1,t}$ and an independent noise vector $\mathbf{v}_t$, $\mathbf{H}_t(\mathbf{x}_{1,t})$, $\mathbf{W}_t(\mathbf{x}_{1,t})$ and $\mathbf{G}_t(\mathbf{x}_{1,t})$ are all known matrices given $\mathbf{x}_{1,t}$, $\mathbf{V}_t$ is a known matrix and $\boldsymbol{\varepsilon}_t$ is an independent noise vector with known Gaussian distribution. In this type of model, given $\mathbf{x}_{1,0:t}$ the conditional model describing the dynamics and the observations for the subset of variables in $\mathbf{x}_{2,t}$ is linear and Gaussian and the density $p(\mathbf{x}_{2,t}|\mathbf{x}_{1,0:t}, \mathbf{y}_{1:t})$ can be computed analytically using the Kalman filter. RBPF algorithms that use Kalman filters to marginalize a subset of state variables of conditionally linear models are often known as mixture Kalman filters (MKFs) [27]. In this case, for each particle $\mathbf{x}_t^{(i)}$ it is necessary to carry over the mean vector and covariance matrix of $p(\mathbf{x}_{2,t}|\mathbf{x}_{1,0:t}^{(i)}, \mathbf{y}_{1:t})$.

2.4.4 Summary

In this chapter we have introduced the Bayesian methodology for target tracking, including the state space models required to describe the dynamics of the target and the measurements related to the target state.

In particular, we have reviewed the most commonly used Bayesian analytic filters for the problem of target tracking. We have presented the Kalman filter (KF), the extended Kalman filter (EKF) and the unscented Kalman filter. These filters are fast and efficient but only work for a subset of state space models and, hence, only a limited number of applications.

We have then described the fundamentals of sequential importance sampling as a numerical solution for the Bayesian filtering problem. The resulting algorithms are often called particle filters. Even though PFs are simulation based methods that provide only an approximate numerical solution, they have the advantage that they can be used with virtually all types of state space models.

We have reviewed some state-of-the-art PFs, including the sequential importance resampling with optimal importance function, the auxiliary particle filter (APF) and the Rao-Blackwellized particle filter.

In the following chapters we are going to make use of these filters in order to solve some specific problems (indoor tracking with RSS and distributed particle filtering).

Table 2.7: Rao-Blackwellized particle filter.

1. Initialization, at $t = 0$:
   - For $i = 1, \ldots, M$, sample $x_0^{(i)} \sim \pi(\mathbf{x}_0)$.
   - For $i = 1, \ldots, M$, set $w_0^{(i)} = \frac{1}{M}$.
2. Recursive step, for $t > 0$:
   - For $i = 1, \ldots, M$, sample $\bar{\mathbf{x}}_{1,t}^{(i)} \sim \pi(\mathbf{x}_{1,t} | \mathbf{x}_{1,0:t-1}^{(i)}, \mathbf{y}_{1:t})$ and set $\mathbf{x}_{1,0:t}^{(i)} \triangleq (\mathbf{x}_{1,0:t-1}^{(i)}, \bar{\mathbf{x}}_{1,t}^{(i)})$.
   - For $i = 1, \ldots, M$, compute $\bar{\mathbf{x}}_{2,t}^{(i)}$ analytically conditioned upon $\bar{\mathbf{x}}_{1,t}^{(i)}$ and set $\mathbf{x}_{2,0:t}^{(i)} \triangleq (\mathbf{x}_{2,0:t-1}^{(i)}, \bar{\mathbf{x}}_{2,t}^{(i)})$.
   - For $i = 1, \ldots, M$, evaluate the weights up to a normalizing constant,
     $$\bar{w}_t^{(i)^*} = w_{t-1}^{(i)^*} p(\mathbf{y}_t | \bar{\mathbf{x}}_t^{(i)}).$$
   - For $i = 1, \ldots, M$, normalize the importance weights,
     $$\bar{w}_t^{(i)} = \frac{\bar{w}_t^{(i)^*}}{\sum_{j=1}^{M} \bar{w}_t^{(j)^*}}.$$
   - Compute $\widehat{M}_{eff}$.
   - If $\widehat{M}_{eff} \leq M_{thres}$ resample:
     - Draw indices $k_1, ..., k_M \in \{1, ..., M\}$ according to the probabilities $\bar{w}_t^{(1)}, ..., \bar{w}_t^{(M)}$.
     - Set $\left(\mathbf{x}_t^{(i)}\right) = \left(\bar{\mathbf{x}}_t^{(k_i)}\right)$ with probability $\bar{w}_t^{(k_i)}$, for $i = 1, ..., M$.
   - Otherwise, set $\left(\mathbf{x}_t^{(i)}\right) = \left(\bar{\mathbf{x}}_t^{(i)}\right)$.

Chapter 3

A multi-model sequential Monte Carlo methodology for indoor tracking

In this Chapter we address the problem of indoor tracking using RSS as a position-dependent data measurement. Since RSS is highly influenced by multipath propagation, it turns out very hard to adequately model the correspondence between the received power and the transmitter-to-receiver distance. Although various models have been proposed in the literature, they often require the use of very large collections of data in order to fit them and display great sensitivity to changes in the radio propagation environment. In this work we advocate the use of switching multiple models that account for different classes of target dynamics and propagation environments and propose a flexible probabilistic switching scheme. The resulting state-space structure is termed a generalized switching multiple model (GSMM) system. Within this framework, we investigate two types of models for the RSS data: polynomial models and classical logarithmic path-loss representation. The first model is more accurate however it demands an offline model fitting step. The second one is less precise but it can be fitted in an online procedure. We have designed two tracking algorithms built around a Rao-Blackwellized particle filter, tailored to the GSMM structure and assessed its performances both with synthetic and experimental measurements.

The remaining of the chapter is organized as follows. In Section 3.1 we present a literature review on the topic. In Section 3.2 we describe the GSMM state space model that represents both the target dynamics

and the associated indoor RSS measurements. In Section 3.3 we give a full account of the experimental setup for the collection of real RSS data and the construction of the observation sub-models. In Section 3.4 we introduce the RBPF and the auxiliary RBPF algorithms for the GSMM system. In Section 3.5 and 3.6 we show numerical and experimental results that illustrate the performance of the method. The chapter is completed with some concluding remarks in Section 3.7.

3.1 Introduction

The problem of indoor tracking has recently received a great deal of attention. The reasons obviously include its many practical applications (e.g., security, guidance, tourism, healthcare, etc [6, 86, 108, 19, 51, 64]) but also the availability of ubiquitous existing communication network infrastructures that can be used for the positioning of mobile terminals. In particular, most current wireless communication networks (WiFi, ZigBee or even cellular networks) provide radio signal strength (RSS) measurements for each radio transmission. The RSS depends on the transmitter-receiver distance and, therefore, it can be used as ranging data for positioning if a suitable model of this dependence is available.

Unfortunately, the construction of such models is far from trivial [18, 62]. Indeed, the RSS is strongly affected by radio propagation phenomena such as scattering or multipath. As a result, RSS measurements are very unstable in practical scenarios. Some authors approach this problem using a class of methods known as fingerprinting [18, 9]. These methods are based on the construction of a radio map that associates directly the RSS measurements to positions on the area of interest. When tracking is performed, new measurements are compared against the RSS values stored in the database, the closest match is selected and the position in the map associated to this match is chosen as the estimate of the target position. These methods require a costly and lengthy calibration procedure and are strongly linked to the specific physical environment. Therefore, if the scenario changes a whole new radio map has to be built.

In order to avoid this burden, other approaches aim at fitting a mathematical model of the data that yields an explicit expression of the RSS measurements as a function of the transmitter-receiver distance. In this case, the usual choice is the classical log-normal path-loss model with attenuation factors that depend on the building floors or building materials [97]. However it has also been shown that the distribution of the RSS

measurements can be non-Gaussian, left-skewed, device-dependent and even multimodal depending on the physical environment [67]. Overall, there is a need for flexible models that can be easily adapted to various kinds of indoor scenarios and handle the effect of scattering or multipath propagation properly.

The design of efficient target tracking algorithms not only requires a proper selection of the model for the observed data, but also the dynamics (i.e. the type of motion) of the target needs to be adequately dealt with [50, 117]. Of special interest in this context is the combination of multiple motion models that has been proposed and investigated in connection with the problem of maneuvering target tracking (see, e.g., [80, 118] and references therein). This type of representation of the target motion has a characteristic switching structure, where a random sequence of indices determines the dynamic regime adopted by the target at every time step [99]. One class of state-space dynamic systems that enables the incorporation of multiple models both for the target motion and the collected observations is the family of so-called jump Markov systems (JMS) [110, 70]. In JMS, the random model indices are assumed to form a Markov chain and every index value determines a model for the target dynamics and an associated model for the observed data.

The assumption of multiple models for the dynamics of the target has an important impact on the design of the tracking algorithm. Interacting multiple model (IMM) algorithms [79] compute a set of estimates of the target state, each one associated to a different dynamic model, and then merge them with appropriate weights in order to obtain a global estimate. Often, the estimates corresponding to each dynamic model are computed using Kalman filters [118, 63]. Recently, however, the sequential Monte Carlo (SMC) methodology (particle filtering) [47, 41, 40, 35] has received considerable attention. In [13], an IMM particle filter is proposed to track a non-Markovian jump system. Other multiple model particle filters have also been proposed in the literature in order to track switching state-space systems. In [20], for instance, each particle is propagated across all possible dynamic models, the importance weights are computed and resampling is performed to (randomly) select the fittest particles and keep their number fixed. A particle filter, for switching observation models has been recently proposed in [23] and exemplified with an outdoor navigation application. Having a "ready-to-use" description of the different physical environments where the tracker has to operate can be advantageous compared to estimating any necessary parameters online. When the environment can be described by sub-models and those are a priori available, the tracking

algorithm only has to detect that the physical scenario has changed and quickly switch to the most useful sub-model. This is in contrast with parameter-estimation based, online techniques, where the adaptation of the parameters involves possibly long transients and also convergence issues[1].

In this chapter, we propose a novel scheme for indoor tracking using RSS that relies on

(a) the representation of both the mobile target dynamics and the resulting RSS observations by means of multiple switching sub-models,

(b) a Rao-Blackwellized particle filter to recursively compute Bayesian estimates of the target position and velocity and

(c) the construction of accurate observation sub-models using experimental data collected by the WSN to be employed for tracking the target.

As for the structure of the state-space system, we introduce a GSMM framework in order to adequately handle the uncertainty in both the dynamics of the target and the RSS observations from the sensors *separately*, in an indoor environment. In particular, we allow the representation of the target motion and the RSS measurements at any time to be drawn from two independent collections of candidate sub-models according to two different indicator random processes (possibly multivariate). Although, strictly speaking, a JMS can be used to represent the same system, using a single indicator to determine the pair of motion and observation models, we advocate the scheme in this paper, with independent indicators, because it is a better fit with our physical intuition (the measurements of the RSS are not necessarily dependent on the type of motion).

The main advantage of the GSMM scheme is that it is flexible enough to encompass a broad range of indoor scenarios. Its main drawback is the need to track the target in a higher dimensional state space (its actual dimension depending on the number of switching sub-models used). We show, however, that the SMC methodology is powerful enough to numerically compute accurate state estimates within this setup. In particular, we propose a RBPF algorithm [50, 27] in which a subset of the state variables, including the observation indicator variables, is integrated out to improve the tracking accuracy. We also propose an implementation of the RBPF algorithm that uses auxiliary variables, in the vein of [94], to sample new particles conditional on the most recent data. The computational complexity of

[1]See [4], Section 4 for a discussion of parameter estimation in the context of particle filtering

the auxiliary RBPF scheme is only marginally higher than that of the RBPF scheme with prior importance distribution and can attain a better performance, specially when the number of particles in the filter is relatively low.

Within the GSMM framework we have studied two types of observation models: polynomial models and logarithmic path-loss models. The first type yields a more precise representation, but requires a very flexible data collection scheme. Its use is intended for offline model construction and assumes that it is possible to collect experimental measurements related to many different transmitter/receiver distances. The logarithmic path-loss models are less accurate, but easier to adjust and require a lesser amount of data. Therefore, we propose their use in an online model construction procedure that can be carried out, automatically, during the startup of the wireless network of sensors. We have fitted models of the two types using a bank of experimental data collected from a network of wireless ZigBee nodes. Then, we have assessed the performance of the proposed RBPF trackers tailored to the resulting GSMM systems both with synthetic and experimental RSS time-series.

3.2 System model

3.2.1 Motion models

Using a state-space formulation, we formally represent the target dynamic state at time $t \in \mathbb{N}$ over a two dimensional region as a 5×1 real vector $\mathbf{x}_{5,t} = [\omega_t, \mathbf{r}_t^\top, \mathbf{v}_t^\top]^\top$, where $\mathbf{r}_t$ represents the target position, $\mathbf{v}_t$ represents the target velocity and ω_t represents a change in the angle of the velocity. Specifically, the position and the velocity contain two real scalars, $\mathbf{r}_t = [r_{1,t}, r_{2,t}]^\top$ and $\mathbf{v}_t = [v_{1,t}, v_{2,t}]^\top$, which are the coordinates of the position, $\mathbf{r}_t$, and velocity, $\mathbf{v}_t$ in the $x - y$ plane while $\omega_t \in \mathbb{R}$ denotes the variation, in radians, of the angle of the velocity at time $t + 1$. The subscript ℓ in $\mathbf{x}_{\ell,t}$ is used to indicate the state vector dimension (this notation will prove useful as we later introduce extended versions of the state vector that incorporate additional variables). We use the "coordinated turn" model [118, 117] presented in Section 2.2.1 to describe the dynamics of $\mathbf{x}_{5,t}$. Specifically, $\mathbf{x}_{5,t}$ evolves with

time according to

$$
\begin{aligned}
\omega_t \quad &\sim \quad p(\omega_t|\omega_{t-1}) \\
\underbrace{\begin{bmatrix} r_{1,t} \\ r_{2,t} \\ v_{1,t} \\ v_{2,t} \end{bmatrix}}_{\mathbf{x}_{4,t}} &= \underbrace{\begin{bmatrix} 1 & 0 & \frac{\sin(\omega_{t-1}T)}{\omega_{t-1}} & -\frac{\cos(\omega_{t-1}T)-1}{\omega_{t-1}} \\ 0 & 1 & \frac{1-\cos(\omega_{t-1}T)}{\omega_{t-1}} & \frac{\sin(\omega_{t-1}T)}{\omega_{t-1}} \\ 0 & 0 & \cos(\omega_{t-1}T) & -\sin(\omega_{t-1}T) \\ 0 & 0 & \sin(\omega_{t-1}T) & \cos(\omega_{t-1}T) \end{bmatrix}}_{\mathbf{A}(\omega_{t-1})} \underbrace{\begin{bmatrix} r_{1,t-1} \\ r_{2,t-1} \\ v_{1,t-1} \\ v_{2,t-1} \end{bmatrix}}_{\mathbf{x}_{4,t-1}} \\
&+ \underbrace{\begin{bmatrix} \frac{1}{2}T^2 & 0 & 0 & 0 \\ 0 & \frac{1}{2}T^2 & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix} u_{1,t} \\ u_{2,t} \\ u_{3,t} \\ u_{4,t} \end{bmatrix}}_{\mathbf{u}_t},
\end{aligned}
\tag{3.1}
$$

where $\mathbf{A}(\omega_{t-1})$ is a transition matrix that depends on the time-discretization period T and the turning angle, $\mathbf{x}_{4,t-1}$ represents the position and velocity in the previous time step, the conditional pdf $p(\omega_t|\omega_{t-1})$ is known and $\mathbf{u}_t$ is a 4×1 real-valued Gaussian vector with zero mean and known covariance matrix, $\mathbf{\Sigma}_u$. As a result, the noise term $\mathbf{Q}\mathbf{u}_t$ is a 4×1 real Gaussian vector, with zero mean and covariance matrix $\mathbf{Q}\mathbf{\Sigma}_u\mathbf{Q}^\top$, that represents the effect of unknown accelerations. Note that the difference between this model and the coordinated turn model introduced in Section 2.2.1 lies in the dynamic nature of the turning angle, ω_t. In Section 2.2.1 the turning angle is fixed in time whilst in this model the angle can vary with time.

By selecting different distributions for ω_t one can devise different motion sub-models. If the target may take any turn at any time, independently of its previous state, then $p(\omega_t|\omega_{t-1}) = p(\omega_t) = U([0, 2\pi))$, where $U(I)$ denotes the uniform density in the interval I. For some vehicles, the turning angle may be constrained to small angles, e.g., $p(\omega_t|\omega_{t-1}) = p(\omega_t) = U([0, \pi/4])$ or even restricted to discrete values, $p(\omega_t|\omega_{t-1}) = p(\omega_t) = U(\{\pm\pi/2\})$. We also note that in the degenerate case in which $\omega_t = 0$ for all t, (3.1) reduces to the so-called "constant velocity" model [50, 117].

Let us assume that, at a given time t, the target of interest may move according to one out of L motion sub-models, indexed by the integers $\{1, 2, \ldots, L\}$. Each motion sub-model corresponds to a different transition pdf for the Markov process $\{\omega_t\}_{t\in\mathbb{N}}$. Therefore, in order to identify the specific densities, we introduce an additional state variable, denoted a_t. This is a discrete random indicator, $a_t \in \{1, \ldots, L\}$, such that $a_{t-1} = l$ implies that ω_t is generated according to the l-th sub-model. Thus, we need to write

$\omega_t \sim p(\omega_t|\omega_{t-1}, a_{t-1})$ to make this dependence explicit. The conditional pmf $p(a_t|a_{t-1})$ is assumed known.

By incorporating the indicator a_t to the target state, we obtain the 6×1 vector $\mathbf{x}_{6,t} = [a_t, \omega_t, \mathbf{r}_t^\top, \mathbf{v}_t^\top]^\top$ which evolves according to the switching multiple model equation

$$\begin{aligned} a_t &\sim p(a_t|a_{t-1}), \\ \omega_t &\sim p(\omega_t|\omega_{t-1}, a_{t-1}), \\ \mathbf{x}_{4,t} &= \mathbf{A}(\omega_{t-1})\mathbf{x}_{4,t-1} + \mathbf{Q}\mathbf{u}_t. \end{aligned} \tag{3.2}$$

3.2.2 Measurement models

We investigate a scheme where RSS observations are collected from J sensors. The measurement provided by the j-th sensor at time t is denoted as $y_{j,t}$. The relationship between the observed RSS, $y_{j,t}$, and the target position, $\mathbf{r}_t$, depends on the physical environment (obstacles, building materials, etc.) [98] and may change with time. In order to handle such uncertainty, we again use a multiple model approach to model the data. Specifically, we allow the observation $y_{j,t}$ to be represented using one out of K different sub-models. This is written as

$$y_{j,t} = f_{m_{j,t}}(\mathbf{r}_t) + \varepsilon_{m_{j,t}}, \tag{3.3}$$

where $m_{j,t} \in \{1, ..., K\}$ is a random index with known pmf $p(m_{j,t})$ that identifies the measurement sub-model at time t for the j-th sensor, $f_{m_{j,t}}$ is the function used to represent the propagation conditions in the $m_{j,t}$-th sub-model, that determine the received RSS, and $\varepsilon_{m_{j,t}} \sim N(\varepsilon_{m_j}; 0, \sigma^2_{m_{j,t}})$ is normally distributed, zero-mean noise with a variance $\sigma^2_{m_{j,t}}$ associated to the $m_{j,t}$-th sub-model and the j-th sensor. The noise contributions are assumed independent across different sensors. The form of each element in the collection of functions $\{f_1, f_2, \ldots, f_K\}$ and the variances $\{\sigma_1^2, \sigma_2^2, \ldots, \sigma_K^2\}$ should be determined from field measurements collected in the scenarios where the tracking system may have to operate (which can be significantly different). See Section 3.3 for further details and examples. This RSS model is very similar to the RSS model introduced in Section 2.2.1 however, in the current model, the function, $f_{m_{j,t}}$, and the noise, $\varepsilon_{m_{j,t}}$ both depend on the selected sensor model, m_j,

We write the measurement-model indicators together in the $J \times 1$ vector $\mathbf{m}_t = [m_{1,t}, \ldots, m_{J,t}]^\top$, hence the full target state has $J + 6$ components, $\mathbf{x}_{J+6,t} = [\mathbf{m}_t^\top, a_t, \omega_t, \mathbf{r}_t^\top, \mathbf{v}_t^\top]^\top$. The observations are put together in the $J \times 1$

vector $\mathbf{y}_t = [y_{1,t}, \ldots, y_{J,t}]^\top$. The indices in $\mathbf{m}_t$ are assumed independent, but not necessarily identically distributed.

3.2.3 Summary

The generalized switching multiple model (GSMM) state-space model that comprises L possible switching sub-models in the state equation and K sub-models in the observation equation is described by the set of relationships

$$\begin{aligned} \mathbf{m}_t &\sim p(\mathbf{m}_t), \\ a_t &\sim p(a_t|a_{t-1}), \\ \omega_t &\sim p(\omega_t|\omega_{t-1}, a_{t-1}), \\ \mathbf{x}_{4,t} &= \mathbf{A}(\omega_{t-1})\mathbf{x}_{4,t-1} + \mathbf{Q}\mathbf{u}_t, \\ \mathbf{y}_t &= \mathbf{f}_{\mathbf{m}_t}(\mathbf{r}_t) + \boldsymbol{\varepsilon}_{\mathbf{m}_t}, \end{aligned} \tag{3.4}$$

where $\mathbf{f}_{\mathbf{m}_t}(\mathbf{r}_t) = \left[f_{m_{1,t}}(\mathbf{r}_t), f_{m_{2,t}}(\mathbf{r}_t), \ldots, f_{m_{J,t}}(\mathbf{r}_t)\right]^\top$ and $\boldsymbol{\varepsilon}_{\mathbf{m}_t} = [\varepsilon_{m_{1,t}}, \ldots, \varepsilon_{m_{J,t}}]^\top$, together with the prior pdf's $p(\omega_0)$, $p(\mathbf{r}_0)$ and $p(\mathbf{v}_0)$ and the pmf $p(a_0)$. Note that the described system model is similar to a jump Markov system [99] however in a JMS the dynamic and the observation equations are controlled by the same model whilst in our description each one is governed by a different model indicator variable (a_t and $\mathbf{m}_t$) that evolves in time according to a different function. In the specific case where $a_t = m_{j,t}$ for all $j = 1, \ldots, J$, our model would become a JMS. For the indoor tracking application, of interest in this paper, the goal is to accurately estimate the sequence of target positions, $\mathbf{r}_{0:t} = \{\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_t\}$, as time evolves.

3.3 Construction of observation models

3.3.1 Experimental scenario and data

We have carried out experiments in a network consisting of nine nodes located at fixed positions (denoted anchors), acting as RSS sensors, and one extra node acting as the moving target (mobile). There is always direct line of sight (LOS) between all pairs of nodes (both anchor-to-anchor and mobile-to-anchor). All the nodes of the wireless sensor network (anchors and mobile) consist of an Arduino board (with an open source prototyping platform) and an XBee (series 1) radio module from Digi, plugged in a daughterboard. The series 1 of XBee are sold as IEEE 802.15.4 OEM modules, as they only support the communication layers, not the upper ZigBee layers from the ZigBee Alliance. We have used two different versions

of these modules, one of them is called XBee and the other one XBee-PRO. The main difference between them is the transmission power range. The XBee sensors are low power modules and their transmission power ranges from -10 to 0 dBm, while the XBee-PRO transmission power varies from 10 to 18 dBm. For our experiments, the XBee nodes were set to 0 dBm and the XBee-PRO nodes to 10 dBm, both of them with an extra added 2 dBi for the omnidirectional antennas to transmit.

The reason for using two kinds of nodes was to observe the influence of the transmission power in real indoor tracking systems. In principle, low power nodes should be more precise for short range tracking whilst high-power nodes should be better for higher ranges. As indoor scenarios are very sensitive to multi-path propagation, we conjectured that mixing the modules would reduce the effect over the whole scenario.

Figure 3.1 illustrates the indoor scenario where we deployed the nine-anchor-node network. The plot on the left shows the deployment of $J = 9$ sensors and their positions in the 6×10 meter area and the plot on the right shows the positions of the mobile sensor when taking static measurements. There are 5 XBee-PRO nodes (depicted in black circles) and 4 XBee nodes (depicted as white circles) and the area covered by the network is of 60 m^2. The positions of the $J = 9$ sensors are $(0, 0)$, $(3, 0)$, $(6, 0)$, $(0, 5)$, $(3, 5)$, $(6, 5)$, $(0, 10)$ $(3, 10)$ and $(6, 10)$, where $(x,\ y)$ denotes horizontal (x) and vertical (y) coordinates, in meters.

All the nodes were configured within the same PAN (personal area network) ID and, to avoid interference with WiFi networks, we set all the nodes in the same channel, number 15, one of the Wi-Fi non-overlapping channels [85]. We used the "IEEE 802.15.4. without ACKs" MAC mode to avoid retransmissions and, consequently, to minimize packet collisions. With this particular transmission mode there is no guarantee that all packets are delivered correctly but, as we configured a small period for packet transmission, we transmitted a sufficiently large number of packets to guarantee the tracking of the mobile node. We assumed that the maximum velocity of the mobile node is 1.3 m/s [17] (simulating a walking person).

In order to build our observation models we collected a large number ($\approx 65,000$) of RSS measurements from each node $j \in \{1, 2, \dots J\}$ at each anchor position. This way we created a database of RSS measurements for each sensor associated to many different anchor-mobile distances. Due to the disposition of each sensor we have a different number of distances for which we have data for each sensor, that is, for sensor $j = 1$ we have measurements at 163 different distances, $d_1^j, d_2^j, \dots, d_{163}^j$, ranging from a minimum of 0.5 m to a maximum of 11.4127 m, whilst for sensor $j = 5$ we have data at 80

different distances ranging from a minimum of 0.5 m to a maximum of 5 m.

As an example, Figure 3.2 shows the raw RSS data collected from sensors 1, 2, 3, 4, 5 and 6. Note the different distances for which we have information and also the difference in the transmitted power from the two sensor types: XBee-Pro (sensors $1, 3, 5$) and XBee (sensors $2, 4, 6$).

We have used all the data collected in the experiments, as shown in Figure 3.2, to construct the polynomial models as they require to collect data related to many different transmitter/receiver distances. Using these data we have constructed $K = 2$ observation sub-models *per sensor*.

For the construction of the logarithmic path-loss models, on the other hand, we only use RSS measurements corresponding to the transmission between pairs of anchor nodes during a short period of time, therefore we only collect data for a small number of different transmitter receiver distances. Figure 3.3 shows the amount of data used in order to build the logarithmic models. For this type of models we have constructed only $K = 2$ sub-models for the *entire network*.

Table 3.1 displays the total number of measurements used and the number of different distances for which we collected RSS data in order to construct each of the observation models. Note that the average number of observations collected for each sensor in order to construct the polynomial models is $\approx 65,000$ whilst the average number of observations collected for the logarithmic models is $\approx 9,000$ per sensor.

3.3.2 Polynomial observation sub-models

In order to construct the $K = 2$ polynomial observation sub-models for each sensor j we have taken the following steps:

1. We have used a k-means algorithm [104] to separate the observations for each distance, d_i^j of every sensor, j, into $K = 2$ clusters, where recall that i is the index used to identify each of the 163 different distances, to which we have data.

2. We have written $S_{i,1}^j$ and $S_{i,2}^j$ to denote the sets of observations assigned to cluster 1 and to cluster 2, respectively, related to the distance d_i^j of sensor j. We have computed the sample mean of each cluster, $S_{i,k}^j$, denoted $\mu_{i,k}^j$, and the sample variance, denoted $\sigma_{i,k}^{j^2}$.

3. We have fitted polynomials of several degrees, $n = 3, 5$ and 7, denoted $g_1^j(u) = \sum_{l=0}^{n} \alpha_{l,1}^j u^l$ and $g_2^j(u) = \sum_{l=0}^{n} \alpha_{l,2}^j u^l$, to the experimentally

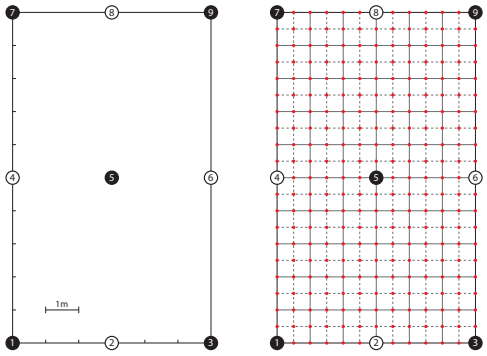


Figure 3.1: Indoor wireless sensor network scenario. The plot in the left shows the deployment of $J = 9$ sensors and their positions in the 6×10 meter area and the plot in the right shows the positions of the mobile sensor when taking static measurements.

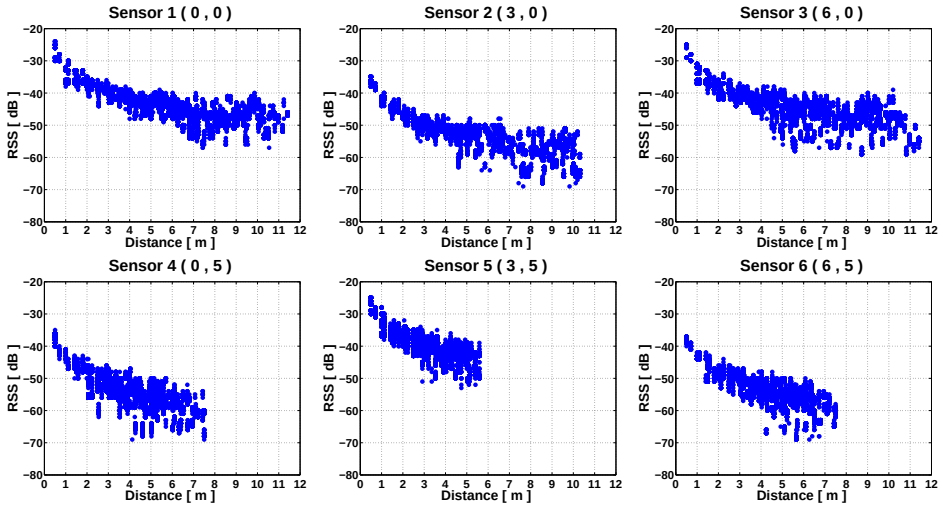


Figure 3.2: Raw RSS data collected by the mobile sensor from anchor sensors 1, 2, 3, 4, 5 and 6. The x-axis shows the transmitter-receiver distance and the y-axis shows the RSS measured in dBs. The title of each graph displays the specific sensor identifier, $j = 1, 2, 3, 4, 5, 6$, and its position in the 6×10 meter area. Note the difference in the transmitted power from the two sensor types: sensors $1, 3, 5$ are XBee-Pro and sensors $2, 4, 6$ are XBee.

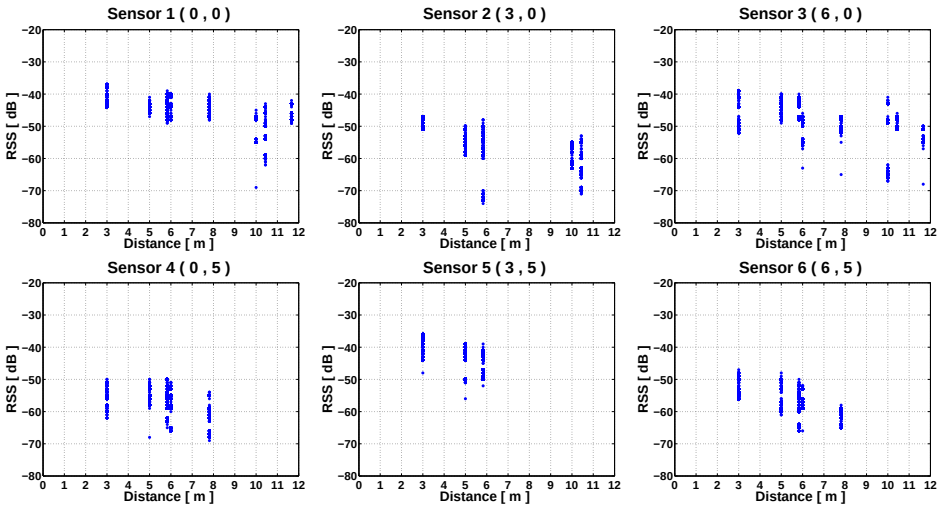


Figure 3.3: Raw RSS data collected from sensors 1, 2, 3, 4, 5 and 6 for the logarithmic models. The x-axis shows the distance among nodes and the y-axis shows the RSS measured in dBs. The title of each graph displays the specific sensor identifier, $j = 1, 2, 3, 4, 5, 6$, and its position in the 6×10 meter area. Note the difference in the transmitted power from the two sensor types: sensors $1, 3, 5$ are XBee-Pro and sensors $2, 4, 6$ are XBee.

Sensor ID	**Pol. # observations**	**Pol. # distances**	**Log. # observations**	**Log. # distances**
1	66,653	163	9,523	8
2	65,205	113	8,463	5
3	66,642	163	9,318	8
4	65,579	77	8,695	5
5	68,774	49	8,315	3
6	65,699	77	8,549	5
7	66,762	163	9,427	8
8	65,533	113	8,652	5
9	66,963	163	9,421	8

Table 3.1: Number of observations and number of different distances to which we collected RSS data in order to construct the polynomial and the logarithmic models.

obtained sequences of sample means $\{\mu_{i,1}^j\}_{i=1}^{I_j}$ and $\{\mu_{i,2}^j\}_{i=1}^{I_j}$, where I_j denotes the number of different distances at which measurements where collected for sensor j. Similarly, we have fitted polynomials of a range of degrees $n = 3, 5$ and 7, denoted $h_1^j(u) = \sum_{l=0}^{n} \beta_{l,1}^j u^l$ and $h_2^j(u) = \sum_{l=0}^{n} \beta_{l,2}^j u^l$, to the sequences of sample variances, $\{{\sigma_{i,1}^j}^2\}_{i=1}^{I_j}$ and $\{{\sigma_{i,2}^j}^2\}_{i=1}^{I_j}$.

With these elements, a preliminary choice of the observation function for the k-th sub-model and the j-th sensor is $f_k^j(\mathbf{r}_t) = g_k^j(d_{j,t})$, where $d_{j,t} = \|\mathbf{r}_t - \mathbf{s}_j\|$ is the distance between the target position, $\mathbf{r}_t$, and the j-th node location, $\mathbf{s}_j$, while the observation noise is Gaussian with zero mean and variance $h_k^j(d_{j,t})$.

Figure 3.4 shows the fitted mean and standard deviation of sensor 1 for models 1 and 2, together with the points from which they are fitted. The depicted polynomials $g_1^1(d)$ and $g_2^1(d)$ that describe the mean and the polynomials $h_1^1(d)$ and $h_2^1(d)$ that describe the variance (depicted here as standard deviation) are polynomials of order $n = 3, 5$ and 7.

With these observation functions, the likelihood of $\mathbf{r}_t$ for each of the individual observations can be defined as

$$p(y_{j,t}|\mathbf{r}_t, m_{j,t}) = N\left(y_{j,t}; g_m^j(d_{j,t}), h_m^j(d_{j,t})\right). \tag{3.5}$$

As the algorithm is intended for indoor application, which limits the tracking area considerably, we have decided to incorporate this information by way of an alternative (truncated) likelihood model,

$$p(y_{j,t}|\mathbf{r}_t, m_{j,t}) \propto \begin{cases} N\left(y_{j,t}; g_m^j(d_{j,t}), h_m^j(d_{j,t})\right), & \text{if } \mathbf{r}_t \in \mathcal{A} \\ 0, & \text{if } \mathbf{r}_t \notin \mathcal{A} \end{cases} \tag{3.6}$$

where $\mathcal{A}$ is the region where the mobile node is allowed to move.

In Section 3.5 we show performance results using polynomial models for the two different likelihood definitions of Eqs.(3.5) and (3.6).

3.3.3 Logarithmic path-loss observation sub-models

The approach to model construction of Section 3.3.2 has some drawbacks. First, we have used a large amount of data, collected prior to the network normal operation by measuring the received power at a fine grid of selected

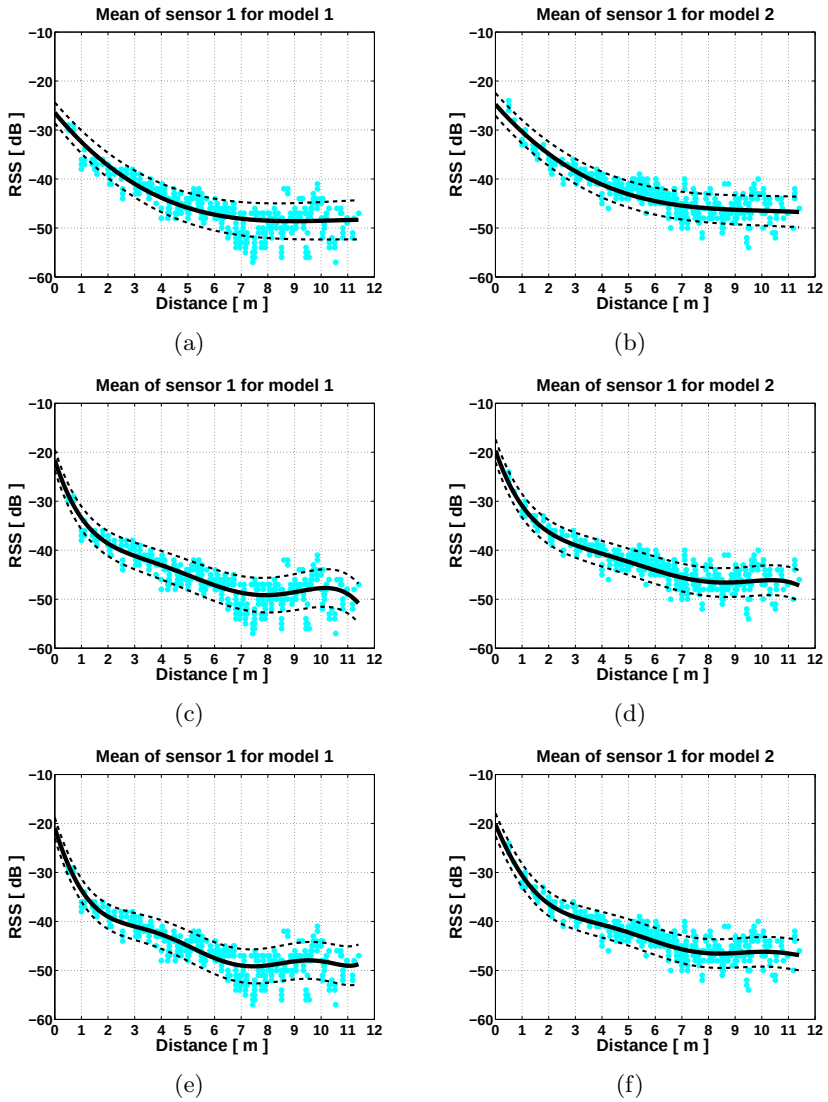


Figure 3.4: Mean (solid line) and standard deviation (dashed line) of sensor 1. The scattered dots represent the raw data. The fitted mean comes from polynomials $g_1^1(d)$ and $g_2^1(d)$ and the standard deviation has been taken from from polynomials $h_1^1(d)$ and $h_2^1(d)$. Figure (a) corresponds to model $m_{1,t} = 1$ and polynomial order $n = 3$, Figure (b) corresponds to model $m_{1,t} = 2$ and polynomial order $n = 3$, Figure (c) corresponds to model $m_{1,t} = 1$ and polynomial order $n = 5$, Figure (d) corresponds to model $m_{1,t} = 2$ and polynomial order $n = 5$, Figure (e) corresponds to model $m_{1,t} = 1$ and polynomial order $n = 7$, Figure (f) corresponds to model $m_{1,t} = 2$ and polynomial order $n = 7$.

points. As a consequence, the model construction procedure is inherently offline and requires some "manual" work.

For a practical implementation we may a prefer a collection of simpler sub-models that can be automatically constructed when the network is started, with a minimum of human intervention. In this section we propose an automatic procedure to construct $K = 2$ observation sub-models using only RSS measurements associated to the messages that are exchanged by the nodes during the network startup. This feature limits considerably the number of different distances for which measurements are available.

Let us consider the automatic definition of $K = 2$ logarithmic observation sub-models for the whole network. When the network is started, the nodes exchange messages and it is possible to record an RSS measurement for each message. Therefore, if there are m_1 pairs of nodes separated by a distance d_i and each pair exchanges m_2 messages, we have $m_1 m_2$ measurements associated to the distance d_i. The first step in building the logarithmic models is, again, to apply a k-means algorithm to the complete set of observations available for each distance d_i. As a result, we obtain two clusters of points, denoted $S_{i,1}$ and $S_{i,2}$, and we need to adjust a parametric model for each cluster.

For simplicity, assume that we collect data corresponding to l different distances and build a single set of observations $S_i = \{s_{i,n}\}_{n=1}^{k}$ for each distance d_i, $i = 1, ..., l$ (independently of the sensors where the data are collected). Let us then consider the problem of using these data to fit the sub-model m,

$$y_{j,t} = f_m(\mathbf{r}_t) + \varepsilon_m = L_{0,m} + \gamma_m 10 \log_{10}\left(\frac{d_0}{d_{j,t}}\right) + \varepsilon_m, \tag{3.7}$$

where $d_{j,t} = ||\mathbf{r}_t - \mathbf{s}_j||$ is the distance between the position of sensor j, and the target at time t; d_0 is a known reference distance; $L_{0,m}$ is the path loss at the reference distance; γ_m is the path loss exponent and $\varepsilon_m \sim N(\varepsilon_m; 0, \sigma^2_{\varepsilon_m})$ is normally distributed, zero-mean noise with variance $\sigma^2_{\varepsilon_m}$. The parameters $L_{0,m}$, γ_m and $\sigma^2_{\varepsilon_m}$ have to be fitted from the experimental data in S_1, ..., S_l.

We propose to select the values of $L_{0,m}$ and γ_m that minimize the mean square error, i.e.,

$$\left(\hat{L}_{0,m}, \hat{\gamma}_m\right) = \arg \min_{L_{0,m},\gamma_m} \left\{ J\left(L_{0,m}, \gamma_m\right) = \sum_{i=1}^{l} \sum_{n=1}^{k} \left(s_{i,n} - L_{0,m} - \gamma_m f_i\right)^2 \right\} \tag{3.8}$$

where $f_i = 10 \log_{10}\left(\frac{d_0}{d_i}\right)$. The cost function $J\left(L_{0,m}, \gamma_m\right)$ is quadratic and,

hence, the problem (3.8) admits a closed-form solution. In particular by taking partial derivatives with respect to $L_{0,m}$ and γ_m and then equating them to zero one finds that

$$\begin{aligned}\hat{\gamma}_m &= \frac{\sum_{j=1}^{l}\sum_{n=1}^{k} s_{j,n}\theta_j}{k\sum_{j=1}^{l} f_j\theta_j},\\ \hat{L}_{0,m} &= \frac{1}{l}\frac{1}{k}\sum_{i=1}^{l}\sum_{n=1}^{k}\left(s_{i,n} - f_i\frac{\sum_{n=1}^{k}\sum_{j=1}^{l} s_{j,n}\theta_j}{k\sum_{j=1}^{l} f_j\theta_j}\right),\end{aligned} \tag{3.9}$$

where $\theta_j = f_i - \frac{1}{l}\sum_{i=1}^{l} f_i$. Finally, the variance $\hat{\sigma}_\varepsilon^2$ can be computed as

$$\hat{\sigma}_\varepsilon^2 = \frac{1}{l}\frac{1}{k}\sum_{i=1}^{l}\sum_{n=1}^{k}\left(s_{i,n} - \hat{L}_{0,m} - \hat{\gamma}_m f_i\right)^2. \tag{3.10}$$

Figure 3.5 shows the fitted mean and standard deviation for models 1 and 2 together with the points from which they are fitted. Note that the amount of data points used in this procedure is considerably less than the amount used to fit the polynomial models (see Figure 3.4).

The simpler version of the likelihood of the logarithmic observation models can be defined as

$$p(y_{j,t}|\mathbf{r}_t, m_t) = N(y_{j,t}; f_m(\mathbf{r}_t), \sigma_{\epsilon,m}), \tag{3.11}$$

where $f_m(\mathbf{r}_t)$ is the function specified in (3.7) and the parameters $L_{0,m}$, γ_m and $\sigma_{\varepsilon,m}^2$ are selected as given by (3.9) and (3.10).

Similar to Section 3.3.2, we modify Eq.(3.11) in order to take into account the area $\mathcal{A}$ where the target is allowed to move. Hence we obtain the truncated Gaussian likelihood

$$p(y_{j,t}|\mathbf{r}_t, m_{j,t}) \propto \begin{cases} N(y_{j,t}; f_m(d_{j,t}), \sigma_{\epsilon,m}), & \text{if } \mathbf{r}_t \in \mathcal{A} \\ 0, & \text{if } \mathbf{r}_t \notin \mathcal{A}. \end{cases} \tag{3.12}$$

3.4 Tracking algorithms

3.4.1 A Rao-Blackwellized particle filter for multiple models

For the estimation of $\mathbf{r}_{0:t}$, we aim at approximating the distribution with pdf

$$p(\mathbf{r}_{0:t}, \omega_{0:t}, a_{0:t}|\mathbf{y}_{1:t}) = \sum_{\mathbf{m}_{0:t}} \int_{\mathbf{v}_{0:t}} p(\mathbf{x}_{J+4,0:t}|\mathbf{y}_{1:t}) d\mathbf{v}_{0:t}. \tag{3.13}$$

Note that since we are interested in tracking $\mathbf{r}_t$ alone, the natural choice of our objective pdf of interest should have been $p(\mathbf{r}_{0:t}|\mathbf{y}_{1:t})$. However, working with the latter marginal density leads to an exponential growth (with t) in the complexity of the SMC algorithm, unless (possibly rough) approximations are used. As our system model allows us to do so, we use the Rao-Blackwellization technique explained in Section 2.4.3 to integrate out $\mathbf{v}_{0:t}$ and $\mathbf{m}_{0:t}$. Specifically, the system model describing the dynamics and observations of $\mathbf{v}_t$ is a conditionally linear Gaussian state-space model, therefore $\mathbf{v}_t$ can be marginalized out and its conditional posterior distribution computed analytically using a Kalman filter (see the Appendix A for details on the recursive computation of $p(\mathbf{v}_t|\mathbf{r}_{0:t-1}, \mathbf{y}_{1:t})$). On the other hand, the random model indicator $\mathbf{m}_t$ is discrete and finite and therefore can also be computed analytically (see Section 3.4.2).

The density of (3.13) cannot be obtained analytically and thus we aim at numerically approximating it with a particle filter. Specifically, we aim at building a point-mass approximation of the distribution with density $p(\mathbf{r}_{0:t}, \omega_{0:t}, a_{0:t}|\mathbf{y}_{1:t})$, using a set of M random samples in the space of $\{\mathbf{r}_{0:t}, \omega_{0:t}, a_{0:t}\}$, denoted $\{\mathbf{r}_{0:t}^{(i)}, \omega_{0:t}^{(i)}, a_{0:t}^{(i)}\}_{i=1}^{M}$, and associated importance weights, $\{w_t^{(i)}\}_{i=1}^{M}$. The pairs $\left\{\left(\mathbf{r}_{0:t}^{(i)}, \omega_{0:t}^{(i)}, a_{0:t}^{(i)}\right), w_t^{(i)}\right\}$, $i = 1, \ldots, M$ are termed particles and we can use them to build the random measure

$$p_M(\mathbf{r}_{0:t}, \omega_{0:t}, a_{0:t}|\mathbf{y}_{1:t}) = \sum_{i=1}^{M} \delta_i(\mathbf{r}_{0:t}, \omega_{0:t}, a_{0:t}) w_t^{(i)}, \tag{3.14}$$

where δ_i is the unit delta measure located at $\left(\mathbf{r}_{0:t}^{(i)}, \omega_{0:t}^{(i)}, a_{0:t}^{(i)}\right)$ and the weights are assumed normalized, i.e., $\sum_{i=1}^{M} w_t^{(i)} = 1$. If the approximation is properly constructed, meaning that the moments of $p_M(\mathbf{r}_{0:t}, \omega_{0:t}, a_{0:t}|\mathbf{y}_{1:t})$ converge to those of $p(\mathbf{r}_{0:t}, \omega_{0:t}, a_{0:t}|\mathbf{y}_{1:t})$ in some adequate sense [32, 72], then it is straightforward to use (3.14) in order to approximate any estimators of $\mathbf{r}_{0:t}$ or $\mathbf{r}_t$. In particular, since

$$\begin{aligned} p_M(\mathbf{r}_t|\mathbf{y}_{1:t}) &= \sum_{a_{0:t}} \int_{\omega_{0:t}} \int_{\mathbf{r}_{0:t-1}} p_M(\mathbf{r}_{0:t}, \omega_{0:t}, a_{0:t}|\mathbf{y}_{1:t}) d\mathbf{r}_{0:t-1} d\omega_{0:t} \\ &= \sum_{i=1}^{M} \delta_i(\mathbf{r}_t) w_t^{(i)}, \end{aligned} \tag{3.15}$$

where δ_i is the delta measure located at $\mathbf{r}_t^{(i)}$, we readily calculate the

(approximate) minimum mean square error (MMSE) estimate of $\mathbf{r}_t$ as

$$\begin{aligned}\hat{\mathbf{r}}_t^{mmse} &= \int \mathbf{r}_t p_M(\mathbf{r}_t|\mathbf{y}_{1:t})d\mathbf{r}_t \\ &= \sum_{i=1}^{M} \mathbf{r}_t^{(i)} w_t^{(i)}. \end{aligned} \tag{3.16}$$

Particles can be generated with a variety of proposal distributions or importance functions (as explained in Section 2.3.3) as long as the weights are computed accordingly. If we choose an importance function that can be factorized as

$$\pi(\mathbf{r}_{0:t}, \omega_{0:t}, a_{0:t}) \propto \pi(\mathbf{r}_t, \omega_t, a_t)\pi(\mathbf{r}_{0:t-1}, \omega_{0:t-1}, a_{0:t-1}), \tag{3.17}$$

we can implement the importance sampling methodology *sequentially*, with a fixed complexity independent of time [41]. A straightforward application of Bayes' theorem yields the recursive decomposition

$$\begin{aligned}p(\mathbf{r}_{0:t}, \omega_{0:t}, a_{0:t}|\mathbf{y}_{1:t}) \propto\ & p(\mathbf{y}_t|\mathbf{r}_t)p(\mathbf{r}_t|\mathbf{r}_{0:t-1}, \omega_{0:t-1})p(\omega_t|\omega_{t-1}, a_{t-1}) \\ & \times p(a_t|a_{t-1})p(\mathbf{r}_{0:t-1}, \omega_{0:t-1}, a_{0:t-1}|\mathbf{y}_{1:t-1}). \end{aligned} \tag{3.18}$$

Proper importance weights can be computed as [77]

$$w_t^{(i)} = \frac{p(\mathbf{r}_{0:t}^{(i)}, \omega_{0:t}^{(i)}, a_{0:t}^{(i)}|\mathbf{y}_{1:t})}{\pi(\mathbf{r}_{0:t}^{(i)}, \omega_{0:t}^{(i)}, a_{0:t}^{(i)})} \tag{3.19}$$

and substituting (3.17) and (3.18) into (3.19) yields the recursive update rule,

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(\mathbf{y}_t|\mathbf{r}_t^{(i)})p(\mathbf{r}_t^{(i)}|\mathbf{r}_{0:t-1}^{(i)}, \omega_{0:t-1}^{(i)})p(\omega_t^{(i)}|\omega_{t-1}^{(i)}, a_{t-1}^{(i)})p(a_t^{(i)}|a_{t-1}^{(i)})}{\pi(\mathbf{r}_t^{(i)}, \omega_t^{(i)}, a_t^{(i)})} \tag{3.20}$$

Moreover, (3.17) means that, at time t, we can draw

$$\left(\mathbf{r}_t^{(i)}, \omega_t^{(i)}, a_t^{(i)}\right) \sim \pi(\mathbf{r}_t, \omega_t, a_t), \quad i = 1, ..., M, \tag{3.21}$$

and append the new samples to the existing streams, $\mathbf{r}_{0:t-1}^{(i)}$, $\omega_{0:t-1}^{(i)}$ and $a_{0:t-1}^{(i)}$ (which need not be modified), to build the sequences $\mathbf{r}_{0:t}^{(i)}$, $\omega_{0:t}^{(i)}$ and $a_{0:t}^{(i)}$. Eqs. (3.21) and (3.20) together yield a SIS type of algorithm for the construction of $p_M(\mathbf{r}_{0:t}, \omega_{0:t}, a_{0:t}|\mathbf{y}_{1:t})$ [41]. Note that (3.18) holds for the system model given in (3.4), where the observations $\mathbf{y}_t$ are conditionally independent given

the position $\mathbf{r}_t$ and the position $\mathbf{r}_t$ is independent of the model indices $a_{0:t-1}$ given the turning angles $\omega_{0:t-1}$.

In order to avoid the degeneracy of the algorithm (see Section 2.3.3 for a discussion of the degeneracy of the SIS methodology), we compute the effective sample size $\hat{M}_{eff} = \frac{1}{\sum_{i=1}^{M} w_t^{(i)2}}$ every time step and each time the effective sample size, $\hat{M}_{eff}$, falls below the resampling threshold, M_{thres} (where $M_{thres} = \lambda M$ for some $0 < \lambda < 1$) we perform resampling.

3.4.2 Evaluation of the weights

In order to ensure that the weights of (3.20) can be computed for an arbitrary importance function $\pi(\mathbf{r}_t, \omega_t, a_t)$, we must be able to evaluate the factors $p(a_t|a_{t-1})$, $p(\omega_t|\omega_{t-1}, a_{t-1})$, $p(\mathbf{r}_t|\mathbf{r}_{0:t-1}, \omega_{0:t-1})$ and $p(\mathbf{y}_t|\mathbf{r}_t)$. The transition densities $p(a_t|a_{t-1})$ and $p(\omega_t|\omega_{t-1}, a_{t-1})$ are part of the model, hence known by assumption. The prior density of the position at time t, $p(\mathbf{r}_t|\mathbf{r}_{0:t-1}, \omega_{0:t-1})$, is Gaussian and can be obtained in closed form for each particle. Indeed, given $\mathbf{r}_{0:t-1}^{(i)}$ and $\omega_{0:t-1}^{(i)}$, the system

$$\begin{bmatrix} r_{1,t}^{(i)} \\ r_{2,t}^{(i)} \\ v_{1,t} \\ v_{2,t} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \frac{\sin(\omega_{t-1}^{(i)}T)}{\omega_{t-1}^{(i)}} & -\frac{\cos(\omega_{t-1}^{(i)}T)-1}{\omega_{t-1}^{(i)}} \\ 0 & 1 & \frac{1-\cos(\omega_{t-1}^{(i)}T)}{\omega_{t-1}^{(i)}} & \frac{\sin(\omega_{t-1}^{(i)}T)}{\omega_{t-1}^{(i)}} \\ 0 & 0 & \cos(\omega_{t-1}^{(i)}T) & -\sin(\omega_{t-1}^{(i)}T) \\ 0 & 0 & \sin(\omega_{t-1}^{(i)}T) & \cos(\omega_{t-1}^{(i)}T) \end{bmatrix} \begin{bmatrix} r_{1,t-1}^{(i)} \\ r_{2,t-1}^{(i)} \\ v_{1,t-1} \\ v_{2,t-1} \end{bmatrix} + \mathbf{Q}\mathbf{u}_t \quad (3.22)$$

is linear and Gaussian, with known parameters, and all posterior pdf's, including $p(\mathbf{r}_t|\mathbf{r}_{0:t-1}^{(i)}, \omega_{0:t-1}^{(i)})$, are Gaussian and can be computed exactly using a Kalman filter [27, 50]. In the sequel, we will denote

$$p(\mathbf{r}_t|\mathbf{r}_{0:t-1}^{(i)}, \omega_{0:t-1}^{(i)}) = N(\mathbf{r}_t; \overline{\mathbf{r}}_{t|t-1}^{(i)}, \overline{\mathbf{\Sigma}}_{t|t-1}^{(i)}). \quad (3.23)$$

See the Appendix A for details on the recursive computation of the mean $\overline{\mathbf{r}}_{t|t-1}^{(i)}$ and the covariance matrix $\overline{\mathbf{\Sigma}}_{t|t-1}^{(i)}$.

The pdf $p(\mathbf{y}_t|\mathbf{r}_t)$ is usually referred to as the likelihood of $\mathbf{r}_t$. If we write $p(y_{j,t}|\mathbf{r}_t)$ as a marginal of the joint density $p(y_{j,t}, m_{j,t}|\mathbf{r}_t)$, then we obtain the expression

$$p(\mathbf{y}_t|\mathbf{r}_t) = \prod_{j=1}^{J} p(y_{j,t}|\mathbf{r}_t) = \prod_{j=1}^{J} \sum_{m_{j,t}=1}^{K} p(y_{j,t}|\mathbf{r}_t, m_{j,t}) p(m_{j,t}), \quad (3.24)$$

where both

$$p(y_{j,t}|\mathbf{r}_t, m_{j,t}) = N(y_{j,t}; f_{m_{j,t}}(\mathbf{r}_t), \sigma^2_{m_{j,t}}) \tag{3.25}$$

and $p(m_{j,t})$ are known from the model, for all $j = 1, ..., J$. See Appendix B for the derivation of (3.24).

3.4.3 Importance functions

A good deal of the performance of the proposed algorithm depends on the choice of the importance function. The simplest choice is the prior

$$\pi(\mathbf{r}_t, a_t, \omega_t) = p(\mathbf{r}_t|\mathbf{r}_{0:t-1}, \omega_{0:t-1})p(\omega_t|\omega_{t-1}, a_{t-1})p(a_t|a_{t-1}), \tag{3.26}$$

which reduces the importance weight calculation to $w_t^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{y}_t|\mathbf{r}_t^{(i)})$.

Table 3.2 shows a summary of the proposed RBPF tracking algorithm, with prior importance function, for the GSMM state-space model.

3.4.4 An auxiliary particle filter for multiple models

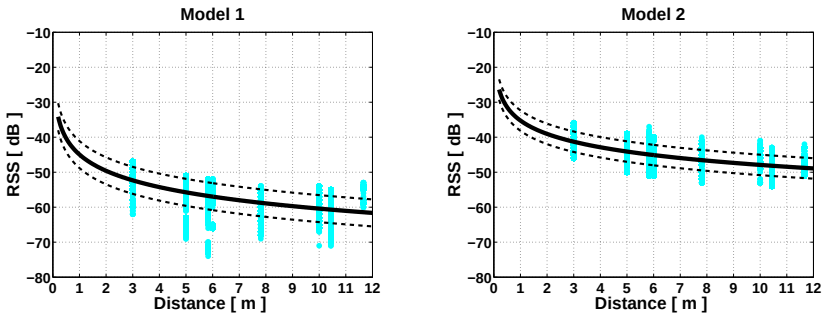
As explained in Section 2.4, more sophisticated importance functions can lead to more efficient algorithms. In this work we investigate the use of an auxiliary SIR algorithm (as the one described in Section 2.4.2) for tracking under the generalized switching multiple model regime. Since the A-SIR filter was already introduced in Section 2.4.2 for a general state-space system, we do not described it again in detail. Here we show how to adapt the A-SIR to the GSMM framework, where we have integrated out some state variables (namely, the observation sub-model indicators, $\mathbf{m}_{0:t}$, and the velocity, $\mathbf{v}_{0:t}$).

To be specific, let $\{\bar{\mathbf{r}}_{0:t}^{(i)}, \bar{\omega}_{0:t}^{(i)}, \bar{a}_{0:t}^{(i)}\}_{i=1}^{M}$ be the particles available at time t before resampling, with importance weights denoted $\bar{w}_t^{(i)}$, $i = 1, ..., M$. We propose to take into account the observation vector $\mathbf{y}_{t+1}$ in the resampling step at time t. This is done by constructing normalized auxiliary weights of the form

$$\lambda_t^{(i)} \propto \bar{w}_t^{(i)} p(\mathbf{y}_{t+1}|\overline{\mathbf{r}}_{t+1|t}^{(i)}), \tag{3.27}$$

for $i = 1, ..., M$, where $\overline{\mathbf{r}}_{t+1|t}^{(i)}$ is the mean of the Gaussian pdf $p(\mathbf{r}_{t+1}|\bar{\mathbf{r}}_{0:t}^{(i)}, \bar{\omega}_{0:t}^{(i)})$, equivalent to Eq. (3.23). Then, we perform importance resampling according to the auxiliary weights, i.e., we draw

$$\left(\mathbf{r}_{0:t}^{(i)}, \omega_{0:t}^{(i)}, a_{0:t}^{(i)}\right) = \left(\bar{\mathbf{r}}_{0:t}^{(k)}, \bar{\omega}_{0:t}^{(k)}, \bar{a}_{0:t}^{(k)}\right) \tag{3.28}$$



(a) (b)

Figure 3.5: Mean and standard deviation of the logarithmic observation model. Figure (a) corresponds to model $m = 1$ and Figure (b) to model $m = 2$. We use the same pair of models for every sensor in the network.

Table 3.2: Rao-Blackwellized particle filter for the GSMM system.

1. Initialization, at $t = 0$:
   - For $i = 1, \ldots, M$, draw a_0, ω_0 and $\mathbf{r}_0$ from the priors $p(\mathbf{r}_0)$, $p(\omega_0)$ and $p(a_0)$, respectively. Set $w_0^{(i)} = \frac{1}{M}$.
2. Recursive step, for $t > 0$:
   - For $i = 1, \ldots, M$, draw $a_t^{(i)} \sim p(a_t|a_{t-1}^{(i)})$, $\omega_t^{(i)} \sim p(\omega_t|\omega_{t-1}^{(i)}, a_{t-1}^{(i)})$ as defined in (3.4) and draw $\mathbf{r}_t^{(i)} \sim p(\mathbf{r}_t|\mathbf{r}_{0:t-1}^{(i)}, \omega_{0:t-1}^{(i)})$ via the Kalman filter equations given in the Appendix A.
   - For $i = 1, \ldots, M$, update the weights, $w_t^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{y}_t|\mathbf{r}_t^{(i)})$, according to (3.24).
   - Find the effective sample size $\hat{M}_{eff} = 1/\sum_{k=1}^{M} w_t^{(k)^2}$. If $\hat{M}_{eff} < \lambda M$ then resample.

with probability $\lambda_t^{(k)}$, for $i = 1, ..., M$ and $k \in \{1, ..., M\}$ (equivalently, we generate M samples from the discrete distribution given by the probabilities $P\left\{\bar{\mathbf{r}}_{0:t}^{(k)}, \bar{\omega}_{0:t}^{(k)}, \bar{a}_{0:t}^{(k)}\right\} = \lambda_t^{(k)}$, $k = 1, ..., M$). The resulting normalized importance weights, after resampling, are

$$w_t^{(i)} \propto \frac{\bar{w}_t^{(i)}}{\lambda_t^{(i)}} = \frac{1}{p(\mathbf{y}_{t+1}|\bar{\mathbf{r}}_{t+1|t}^{(k_i)})}, \tag{3.29}$$

where $k_i \in \{1, ..., M\}$ is the index such that $(\mathbf{r}_{0:t}^{(i)}, \omega_{0:t}^{(i)}, a_{0:t}^{(i)}) = (\bar{\mathbf{r}}_{0:t}^{(k_i)}, \bar{\omega}_{0:t}^{(k_i)}, \bar{a}_{0:t}^{(k_i)})$ and $\overline{\mathbf{r}}_{t+1|t}^{(k_i)}$ is the mean of the Gaussian density $p(\mathbf{r}_{t+1}|\bar{\mathbf{r}}_{0:t}^{(k_i)}, \bar{\omega}_{0:t}^{(k_i)})$. The weights, $w_t^{(i)}$, and the resampled particles, $(\mathbf{r}_{0:t}^{(i)}, \omega_{0:t}^{(i)}, a_{0:t}^{(i)})$, $i = 1, ..., M$, are carried over to the next iteration of the algorithm.

Table 3.3 shows a summary of the proposed auxiliary RBPF (A-RBPF) algorithm for the proposed GSMM system.

3.4.5 Computational complexity of the algorithms

The computational load of the (basic) RBPF and the A-RBPF algorithms is very similar. From Tables 3.2 and 3.3 it is apparent that, for each particle, the main operations to be carried out are the prediction and update steps of the Kalman filters[2] and the evaluation and multiplication of J Gaussian densities for the computation of the weights. Hence, the complexity of the particle filters, excluding the resampling step, is $\mathcal{O}(MJ)$, i.e., it grows linearly with the number of particles and the number of sensors.

As for the resampling step, we have implemented a multinomial resampling procedure (for both algorithms) which has complexity $\mathcal{O}(M^2)$. This complexity could be reduced to $\mathcal{O}(M \log M)$ using, e.g., systematic resampling (see, e.g., [24] and [38] for details). Overall, the (asymptotic) computational complexity of the two PFs is $\mathcal{O}(M(M + J))$.

The A-RBPF algorithm, however, requires some additional computational effort compared to the basic RBPF method. This is because the A-RBPF algorithm demands that resampling is performed (unconditionally) at every time step, while in the basic RBPF method resampling is only carried out when the effective sample size $\hat{M}_{eff}$ falls below a user-defined threshold. Once the filter is "locked" to the target, $\hat{M}_{eff}$ can possibly stay over the threshold for several time steps in a row.

[2]Note that the Kalman filters are run for a state space model with only two state variables and two observations.

Table 3.3: Auxiliary RBPF algorithm for the GSMM system.

1. Initialization, at $t = 0$:
   - For $i = 1, \ldots, M$, draw a_0, ω_0 and $\mathbf{r}_0$ from the priors $p(\mathbf{r}_0)$, $p(\omega_0)$ and $p(a_0)$, respectively. Set $w_0^{(i)} = \frac{1}{M}$.
2. Recursive step, for $t > 0$:
   - For $i = 1, \ldots, M$, draw $\bar{a}_t^{(i)} \sim p(a_t|a_{t-1}^{(i)})$, $\bar{\omega}_t^{(i)} \sim p(\omega_t|\omega_{t-1}^{(i)}, a_{t-1}^{(i)})$ as defined in (3.4) and draw $\bar{\mathbf{r}}_t^{(i)}$ from the Gaussian pdf $p(\mathbf{r}_t|\mathbf{r}_{0:t-1}^{(i)}, \omega_{0:t-1}^{(i)})$ computed via the Kalman filter equations given in the Appendix A. Let $\bar{\mathbf{r}}_{0:t}^{(i)} = \{\bar{\mathbf{r}}_t^{(i)}, \mathbf{r}_{0:t-1}^{(i)}\}$, $\bar{\omega}_{0:t}^{(i)} = \{\bar{\omega}_t^{(i)}, \omega_{0:t-1}^{(i)}\}$ and $\bar{a}_{0:t}^{(i)} = \{\bar{a}_t^{(i)}, a_{0:t-1}^{(i)}\}$.
   - For $i = 1, \ldots, M$, update the weights, $\bar{w}_t^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{y}_t|\bar{\mathbf{r}}_t^{(i)})$ according to (3.24).
   - Compute the Gaussian predictive distributions $p(\mathbf{r}_{t+1}|\bar{\mathbf{r}}_{0:t}^{(i)}, \bar{\omega}_{0:t}^{(i)}) = N(\mathbf{r}_{t+1}; \overline{\mathbf{r}}_{t+1|t}^{(i)}, \overline{\mathbf{\Sigma}}_{t+1|t}^{(i)})$ using Kalman filtering.
   - Resampling:
     - For $i = 1, ..., M$, compute the normalized auxiliary weights $\lambda_t^{(i)} = \bar{w}_t^{(i)} p(\mathbf{y}_{t+1}|\overline{\mathbf{r}}_{t+1|t}^{(i)})$.
     - Draw indices $k_1, ..., k_M \in \{1, ..., M\}$ according to the probabilities $\lambda_t^{(1)}, ..., \lambda_t^{(M)}$.
     - Set $\left(\mathbf{r}_{0:t}^{(i)}, \omega_{0:t}^{(i)}, a_{0:t}^{(i)}\right) = \left(\bar{\mathbf{r}}_{0:t}^{(k_i)}, \bar{\omega}_{0:t}^{(k_i)}, \bar{a}_{0:t}^{(k_i)}\right)$ with probability $\lambda_t^{(k_i)}$, for $i = 1, ..., M$.
     - Update the weights as $w_t^{(i)} = \frac{1}{p(\mathbf{y}_{t+1}|\overline{\mathbf{r}}_{t+1|t}^{(k_i)})}$, for $i = 1, ..., M$.

We have carried out a simple computer simulation to illustrate the effect the of the different scheduling of the resampling steps on the execution time of the RBPF and the A-RBPF algorithms. In particular, Figure 3.6 shows the average execution time per time step of the two filters versus several values of the number of particles M. For the experiment, the two filters were coded in Matlab and we run eight independent simulations (each one with a different number of particles, M, from 100 to $50,000$) using synthetically generated observations from the $J = 9$ sensors and the two polynomial observation sub-models per sensor. After each simulation, the execution time was recorded and divided by the number of time steps (100 in all simulations).

It can be seen that the execution time of the A-RBPF method is always higher. The difference is negligible for small to medium values of M but becomes significant when the number of particles grows ($M \geq 5,000$). This is as expected, since the extra resampling steps carried out by the A-RBPF scheme have complexity $\mathcal{O}(M^2)$ as discussed above.

In the next section we show performance results (both with synthetic and experimental data) for the two algorithms with the two proposed observation models *and* likelihoods.

3.5 Computer simulations results

In order to illustrate the performance of the proposed methods we have generated synthetic data for three specific test trajectories in an indoor scenario of 6×10 m, shown in Figure 3.7. One of the trajectories is (almost) linear whilst the other two are more irregular. From now on, we denote the trajectory in the first row as Trajectory 1, the trajectory in the middle row as Trajectory 2 and the trajectory in the bottom row as Trajectory 3.

We have also generated random trajectories using the state-space model of (3.2) in order to assess the algorithm performance when we have a complete match between the model and the filters. We have chosen $L = 2$ motion sub-models, corresponding to a CV model, which describes a linear movement, and a CT model, which represents the kinematics of a turn. Hence, the dynamic-model indicator can take two values, $a_t \in \{1, 2\}$. When $a_t = 1$ the CV model is chosen and when $a_t = 2$ the CT model is chosen. As the dynamics of a person in an indoor scenario can be described as mostly linear except for some occasional maneuvers, the probability of the CV model is always higher, specifically, we have chosen $p(a_t = 1|a_{t-1} = 1) = 0.8$ and $p(a_t = 1|a_{t-1} = 2) = 1$. The CV model is obtained by setting $\omega_t = 0$.

Therefore, the transition probability distribution for the angle ω_t in the CV model is, trivially, $p(\omega_t = 0|\omega_{t-1}, a_{t-1} = 1) = 1$ for any ω_{t-1}. The values we allow the turning angle to take with the CT model are $\omega_t \in \{\pm\frac{\pi}{6}, \pm\frac{\pi}{4}, \pm\frac{\pi}{2}, \pm\frac{3\pi}{4}, \pm\pi\}$ and the transition probabilities are uniform, i.e., $p(\omega_t|\omega_{t-1}, a_{t-1} = 2) = \frac{1}{10}$ for all ω_t and ω_{t-1}.

We have then generated four synthetic data sets using the two suggested type of models (the polynomial sub-models described in Section 3.3.2 and the logarithmic sub-models described in Section 3.3.3) with the two suggested likelihoods (the Gaussian likelihoods defined in (3.5) and (3.11), and the truncated Gaussians defined in (3.6) and (3.12)), for all trajectories (the three test trajectories and a random trajectory).

Note that the A-RBPF algorithm uses the truncated likelihoods of (3.6) and (3.12) to compute the weights only. When computing the predictive likelihood $p(\mathbf{y}_{t+1}|\bar{\mathbf{r}}_{t+1|t}^{(k_i)})$ of (3.29) we have used the (simpler) functions of (3.5) and (3.11)[3].

Using these data, we have performed tracking with the two proposed particle filters. Recall that the specific form of the particle filter depends on the assumed model (which determines the likelihood), hence we have run four algorithms. Table 3.4 lists the simulation and algorithm parameter settings for the RBPF and the A-RBPF algorithms.

3.5.1 Order of polynomial models

In order to select a suitable polynomial order we have run 500 independent simulations with both the RBPF and the A-RBPF algorithms with polynomials of three different orders, $n = 3, 5$ and 7. For this particular simulation, we have selected Gaussian likelihoods. We have then evaluated the mean and standard deviation of the absolute error in the estimation of the position with each algorithm. The results are displayed in Table 3.5, where MAE stands for mean absolute error and SDE stands for standard deviation of the error. We have applied the algorithms with $M = 100$ and $M = 500$ particles.

When tracking a random trajectory, the RBPF algorithm with polynomial order $n = 5$ achieves the lowest MAE and the polynomial of order $n = 3$ achieves the lowest SDE. For the same trajectory, the A-RBPF algorithm with polynomial order $n = 7$ achieves the lowest MAE whist the

[3]Note that, when $\bar{\mathbf{r}}_{t+1|t}^{(k_i)} \notin \mathcal{A}$, the truncated likelihoods of (3.6) become a vector with all zero entries and, as a consequence, the weights of Eq.(3.29) would not be properly defined.

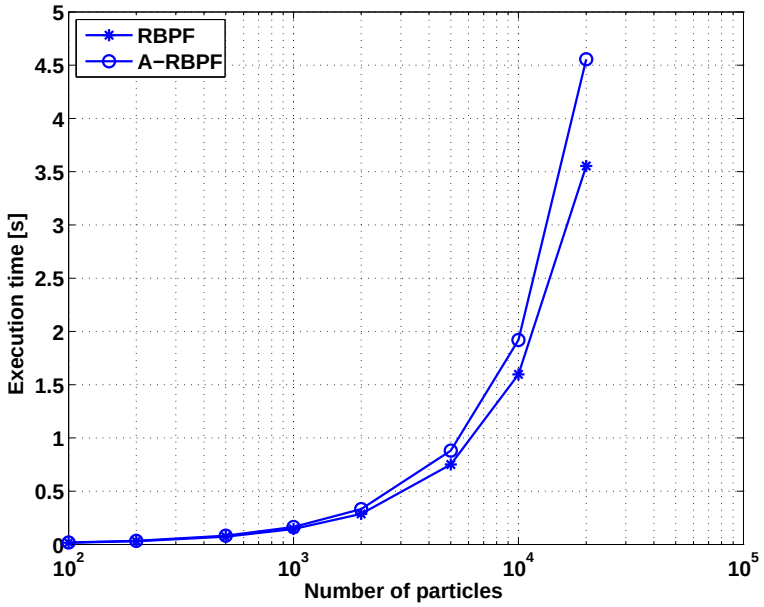


Figure 3.6: Average execution time per time step of the RBPF and the A-RBPF algorithms.

Parameter	**Value**	**Parameter**	**Value**
Total no. of particles, M	100/500	Sampling period, T_s	0.5
Resampling threshold, λ	0.2	Position noise variance, $\sigma^2_{u,12}$	1
Trajectory 1 simulation time, T_1	60 s	Velocity noise variance, $\sigma^2_{u,34}$	0.5
Trajectory 2 simulation time, T_2	50 s	Model 1 probability, $p(m_k{=}1)$	0.4
Trajectory 1 simulation time, T_3	85 s	Model 2 probability, $p(m_k{=}2)$	0.6
Polynomial orders, n	3, 5, 7	No. of independent simulations	500

Table 3.4: Algorithm and parameters table for the RBPF and A-RBPF algorithms.

Algorithm	**Pol. order**	**M**	**Rnd. trajectory**		**Trajectory 1**		**Trajectory 2**		**Trajectory 3**	
			MAE [m]	**SDE [m]**	**MAE [m]**	**SDE [m]**	**MAE [m]**	**SDE [m]**	**MAE [m]**	**SDE [m]**
RBPF	3	100	0.673	0.682	0.493	0.299	0.509	0.312	0.521	0.322
pol. functions	3	500	0.512	0.362	0.474	0.282	0.487	0.298	0.494	0.301
A-RBPF	3	100	0.560	0.469	0.483	0.292	0.503	0.312	0.507	0.312
pol. functions	3	500	0.499	0.336	0.473	0.473	0.485	0.295	0.494	0.302
RBPF	5	100	0.656	0.732	0.441	0.281	0.490	0.313	0.482	0.315
pol. functions	5	500	0.466	0.354	0.416	0.261	0.465	0.292	0.452	0.292
A-RBPF	5	100	0.528	0.483	0.432	0.276	0.481	0.305	0.470	0.308
pol. functions	5	500	0.456	0.334	0.418	0.263	0.467	0.293	0.453	0.294
RBPF	7	100	0.704	0.847	0.434	0.282	0.482	0.307	0.465	0.309
pol. functions	7	500	0.460	0.370	0.409	0.260	0.454	0.285	0.436	0.283
A-RBPF	7	100	0.526	0.504	0.425	0.276	0.469	0.298	0.457	0.303
pol. functions	7	500	0.447	0.335	0.413	0.262	0.454	0.286	0.439	0.285

Table 3.5: Mean of 500 independent simulations for the mean absolute error of the position, in units of meters, and standard deviation of the error for the two proposed algorithms for a random trajectory and for the specified 3 trajectories. The first line corresponds to the results for $M = 100$ particles, the second line for $M = 500$ particles. The used likelihoods are polynomial models with $n = 3, 5$ and 7 polynomial orders.

polynomial of order $n = 5$ achieves the lowest SDE. When tracking the three specific trajectories the errors obtained are similar. Overall, the gain provided by a higher order polynomials is small. Therefore, in order to avoid the risk of over-fitting the models, we have chosen to use polynomials of order $n = 3$ for the rest of the simulations.

3.5.2 Tracking performance

Figure 3.7 illustrates the tracking capacity of the A-RBPF algorithm with 200 particles using the polynomial sub-models (with a polynomial order of $n = 3$) and the logarithmic sub-models for the four test trajectories. All of them have been simulated with Gaussian likelihoods and with truncated Gaussian likelihoods. The true trajectories are shown in solid dark-colored lines and the estimates provided by the particle filter are shown in solid red lines. The locations of the sensors are depicted as black squares.

In each column we show the tracking performance of a different algorithm, starting from the left: column 1 shows the tracking capabilities of an A-RBPF algorithm that uses multiple switching logarithmic sub-models whose likelihoods are Gaussians as the one described in (3.11), column 2 shows the tracking capabilities of an A-RBPF scheme that uses multiple switching polynomial sub-models whose likelihoods are Gaussians as the one described in (3.5), column 3 shows the tracking capabilities of an A-RBPF scheme that uses multiple switching logarithmic sub-models whose likelihoods are truncated Gaussians as the one described in (3.12) and column 4 shows the tracking capabilities of an A-RBPF algorithm that uses multiple switching polynomial sub-models whose likelihoods are truncated Gaussians as the one described in (3.6). All of the algorithms have been tried with 200 particles.

In each row we show the tracking capabilities of the algorithms for a specific trajectory, that is, the first row shows tracking examples of all four algorithms for Trajectory 1, the second row shows tracking examples for Trajectory 2 and the third row shows tracking examples for Trajectory 3.

In order to numerically assess the performance of the proposed PFs we have run 500 independent simulations with each class of observation model (polynomial and logarithmic, Gaussian and truncated Gaussian) and we have evaluated the mean and standard deviation of the absolute error in the estimation of the position with each algorithm. The results for Gaussian likelihoods are displayed in Table 3.6 and the results for truncated Gaussian likelihoods are displayed in Table 3.7. We have applied the algorithms with $M = 100$ and $M = 500$ particles.

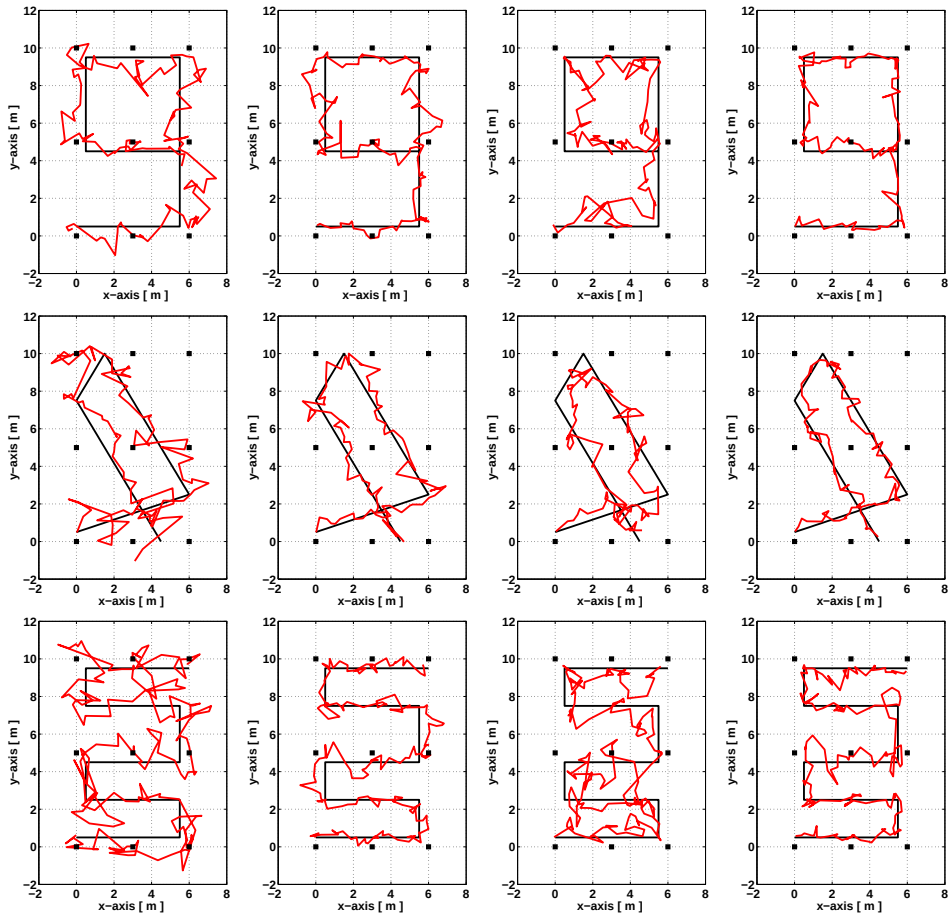


Figure 3.7: Results of target tracking with three different reference trajectories and four types of synthetic observation data, each one created according to the four likelihood functions described in Section 3.3.1. The simulated trajectories are plotted in solid black lines, the estimated trajectories in solid red lines and the sensors are depicted with dark squares. To perform tracking we have used four different algorithms that correspond to an A-RBPF algorithm of 200 particles that uses the four different likelihood functions. In each column we show the tracking performance of a different algorithm and in each row we show the tracking capabilities of the algorithms for a specific trajectory.

Algorithm	M	Rnd. trajectory MAE [m]	Rnd. trajectory SDE [m]	Trajectory 1 MAE [m]	Trajectory 1 SDE [m]	Trajectory 2 MAE [m]	Trajectory 2 SDE [m]	Trajectory 3 MAE [m]	Trajectory 3 SDE [m]
RBPF	100	1.123	0.813	0.994	0.611	1.075	0.653	1.059	0.648
log. functions	500	1.009	0.710	0.935	0.568	1.021	0.611	0.999	0.611
A-RBPF	100	1.057	0.757	0.969	0.599	1.051	0.632	1.037	0.632
log. functions	500	0.999	0.703	0.930	0.566	1.018	0.609	0.996	0.611
RBPF	100	0.734	0.859	0.459	0.282	0.460	0.294	0.502	0.325
pol. functions	500	0.483	0.352	0.441	0.269	0.444	0.283	0.473	0.295
A-RBPF	100	0.540	0.478	0.453	0.278	0.453	0.288	0.485	0.309
pol. functions	500	0.472	0.329	0.443	0.271	0.444	0.282	0.472	0.295
IMM-UKF log. functions		1.534	1.060	1.404	0.900	1.428	0.924	1.397	0.902
IMM-UKF pol. functions		0.686	0.400	0.609	0.360	0.620	0.380	0.620	0.374

Table 3.6: Mean of 500 independent simulations for the mean absolute error of the position, in units of meters, and standard deviation of the error for the two proposed algorithms and the two proposed observation models for a random trajectory and for the specified 3 trajectories. The first line corresponds to the results for $M = 100$ particles, the second line for $M = 500$ particles. The used likelihoods are Gaussians.

Algorithm	M	Rnd. trajectory MAE [m]	Rnd. trajectory SDE [m]	Trajectory 1 MAE [m]	Trajectory 1 SDE [m]	Trajectory 2 MAE [m]	Trajectory 2 SDE [m]	Trajectory 3 MAE [m]	Trajectory 3 SDE [m]
RBPF	100	0.987	0.809	0.769	0.480	0.893	0.504	0.810	0.501
log. functions	500	0.868	0.555	0.741	0.470	0.857	0.482	0.767	0.474
A-RBPF	100	0.885	0.569	0.751	0.473	0.889	0.500	0.790	0.493
log. functions	500	0.857	0.534	0.737	0.469	0.855	0.481	0.766	0.473
RBPF	100	1.500	2.122	0.367	0.244	0.427	0.263	0.407	0.272
pol. functions	500	0.503	0.414	0.349	0.230	0.411	0.250	0.386	0.253
A-RBPF	100	0.503	0.424	0.362	0.236	0.420	0.256	0.400	0.266
pol. functions	500	0.422	0.276	0.350	0.232	0.411	0.250	0.387	0.254

Table 3.7: Mean of 500 independent simulations for the mean absolute error of the position, in units of meters, and standard deviation of the error for the two proposed algorithms and the two proposed observation models for a random trajectory and for the specified 3 trajectories. The first line corresponds to the results for $M = 100$ particles, the second line for $M = 500$ particles. The used likelihoods are truncated Gaussians.

We first note that the algorithms that use polynomial observation sub-models perform better at estimating the position of the target than the algorithms that use logarithmic observation sub-models (both for the RBPF and the A-RBPF algorithms). In Table 3.6 we see that, for Trajectories 1, 2 and 3, the algorithms that use polynomial observation sub-models achieve a MAE of ≈ 0.4 m, with a standard deviation of ≈ 0.25 m for the position, whilst the algorithms that use logarithmic observation sub-models achieve a MAE of ≈ 0.8 m, with a standard deviation of ≈ 0.5 m for the position. These results are as a priori expected, since the polynomial functions are much more elaborate: we employ $m = 2$ observation sub-models per sensor and a larger amount of data has been used to fit these models. The logarithmic functions, on the other hand, are obtained by a simpler procedure, they involve only $m = 2$ observation sub-models for the whole sensor network and they have been fitted with less data.

Tables 3.6 and 3.7 also show that algorithms that use truncated Gaussian likelihoods obtain better results than algorithms that use Gaussian likelihoods. Once again these results are as a priori expected, since we are incorporating information regarding the area where the object is allowed to move. However, in many cases such information may be unavailable or the boundaries may be fuzzy. In that case, we can only resort to the use of Gaussian likelihoods.

Finally, it should be noted that the A-RBPF method is more efficient than the RBPF algorithm as it obtains better estimation accuracy with a lesser number of particles. This is specially noticeable when the particle number is small (e.g., $M = 100$). Therefore, if the algorithm should run with few samples (due to computational complexity constraints), the A-RBPF algorithm should be the technique of choice. If a larger number of particles can be afforded, then both methods achieve a similar performance but the resampling and weight update step of the RBPF scheme are simpler.

3.5.3 Comparison with the interacting multiple model methodology

Table 3.6 also shows a comparison of the proposed RBPF algorithms with a state-of-the-art interacting multiple model (IMM) algorithm [79, 63] that uses a bank of unscented Kalman filters (UKFs) [65, 114].

A brief explanation regarding the structure and complexity of the IMM-UKF tracker is necessary. In the considered GSMM setup, we have $L = 2$ motion sub-models, corresponding to a CV model and a CT model. The latter can take up to 11 different possible turns. In the IMM scheme, this

translates to $\tilde{L} = 12$ different dynamic models. Furthermore, we have $K = 2$ observation sub-models for each of the $J = 9$ sensors, therefore we have $K^J = 512$ combinations of observation models for each time instant. As a consequence, the IMM structure should account for $\tilde{L} \times K^J = 6,114$ filters, each one matched to a different dynamic model and a combination of observation models. Since such implementation is prohibitive for online tracking, we have built a bank of twelve UKFs matched to the $\tilde{L} = 12$ dynamic models. For each UKF, the combination of observation sub-models for the sensors is fixed. In particular, we select it by solving the maximum a posteriori estimation problem

$$\hat{\mathbf{m}}_t = \arg\max_{\mathbf{m}_t} p(\mathbf{m}_t|\mathbf{y}_t, \hat{\mathbf{r}}_t),$$

where

$$\begin{aligned} p(\mathbf{m}_t|\mathbf{y}_t, \hat{\mathbf{r}}_t) &= p(\mathbf{y}_t, \hat{\mathbf{r}}_t|\mathbf{m}_t)p(\mathbf{m}_t) \\ &= N(\mathbf{y}_t; \mathbf{f}(\hat{\mathbf{r}}_t, \mathbf{m}_t), \mathbf{\Sigma}_{\varepsilon}^{\mathbf{m_t}})p(\mathbf{m}_t) \end{aligned} \tag{3.30}$$

is the posterior probability of the sub-model indices and $\hat{\mathbf{r}}_t$ is a prediction of the target position. We obtain the latter as a weighted sum of the predictions of each of the bank of UKFs,

$$\hat{\mathbf{r}}_t = \sum_{l=1}^{\tilde{L}} \mu_{t-1}^{(l)} \hat{\mathbf{r}}_t^{(l)} \tag{3.31}$$

where $\{\mu_{t-1}^l\}_{l=1}^{\tilde{L}}$ are the mixture weights of the $\tilde{L}$ dynamic models of the IMM algorithm [99] and $\hat{\mathbf{r}}_t^{(l)}$ is the prediction of $\mathbf{r}_t$ produced by the l-th UKF in the bank.

The last row of Table 6 displays mean errors (and standard deviations) obtained by the IMM-UKF tracker under the same simulation setup as the proposed RBPF algorithms with Gaussian likelihoods. When using logarithmic observation sub-models, both the RBPF and the A-RBPF algorithms clearly outperform the IMM-UKF tracker. When using polynomial sub-models, the A-RBPF and the RBPF algorithms with 500 particles are also clearly superior to the IMM-UKF tracker. Only the simplest particle filter (the RBPF algorithm) with 100 particles attains a slightly worse performance.

This results are specially relevant if we notice that the IMM-UKF tracker is computationally heavy (each UKF needs 35 σ-points and detects the J model indices in $\mathbf{m}_t$ at each time step). Therefore the A-RBPF tracker with 100 particles is both more accurate and computationally lighter than the IMM-UKF in all the considered setups.

3.5.4 Gain from using multiple models

In order to assess the improvement in the accuracy that we attain with the use of a multiple model description, we have also run simulations with exactly the same settings as before but this time using a *single* logarithmic observation model. To do so, we have constructed an extra observation model following the steps described in Section 3.3.3 but skipping the clustering step, that is, we avoid dividing the data into clusters to construct the logarithmic model. Same as with the other observation models, we have constructed a Gaussian likelihood and a truncated Gaussian likelihood.

Table 3.8 displays the empirical mean and standard deviation of the absolute error in the estimation of the position obtained when we average 500 independent simulations using an RBPF and an A-RBPF algorithms with a single logarithmic model (using both Gaussian and truncated Gaussian likelihoods).

If we compare the results in Table 3.8 to those obtained with the multiple switching logarithmic sub-models in Tables 3.6 and 3.7, we observe that the GSMM approach yields a reduction in the MAE and the SDE of the position of approximately 36% in all algorithms and all trajectories using both types of likelihoods (Gaussians and truncated Gaussians). For example, the average MAE of the position in the random trajectory using the RBPF scheme with multiple switching logarithmic Gaussian sub-models is 1.123 m whilst the RBPF scheme with a single logarithmic Gaussian model obtains 1.788 m. This means that we obtain a reduction in the MAE of the position of 36%.

3.6 Experimental results

One major goal of this work is the experimental verification of the validity of the GSMM scheme and the particle filtering algorithms. Hence, we have also applied the same techniques (with exactly the same parameters) to the three test trajectories using experimentally collected data. The measurements were obtained with the same setup described in Section 3.3.1, but independently from the observations used to fit the models.

One limitation of this experimental data set is that we do not know the exact position of the target node at the time instants when the observations were collected. As a consequence, it is not possible to calculate numerical errors for the target position estimates. We do know the overall trajectory of the target, however, and we can plot it together with the complete track estimate for comparison.

Figure 3.8 shows the tracking performance of the A-RBPF algorithm with 200 particles using polynomial observation models and logarithmic observation models (both for Gaussian and truncated Gaussian likelihoods) when fed with real data. The true trajectories are shown in solid black lines, the estimated trajectories in solid red lines and the sensor positions with black squares.

As we can observe, truncating the likelihood really improves the tracking capabilities, as both observation models (polynomial and logarithmic) improve considerably when using this type of likelihood. When the boundaries of the area to be monitored are not known a priori (hence, there is no possible truncation of the likelihood function), the algorithms that use polynomial models clearly outperform those that employ logarithmic functions (which is as expected, since they have been fitted with more data).

Figure 3.9 shows the tracking performance of the IMM-UKF tracker and the A-RBPF algorithm with $M = 200$ particles and experimental data. All the algorithms use logarithmic observation models with Gaussian likelihoods. The column on the left shows the outcome of the IMM-UKF tracker and the column on the right illustrates the performance of the A-RBPF algorithm. It is observed that the IMM-UKF tracker has difficulties following the three experimental trajectories and that the estimated trajectories are noisier than those of the A-RBPF algorithm.

3.7 Conclusions

We have proposed a generalized switching multiple-model (GSMM) approach to the representation of the target dynamics and the radio signal-strength (RSS) observations in an indoor scenario. The resulting class of state-space models is very flexible and we claim that it may enable the adequate formal representation of time-varying scenarios with highly unstable RSS measurements. The drawback of the GSMM system is the increase in the dimension of the system state and, hence, the number of variables that the tracking algorithm has to estimate. To handle this difficulty, we have introduced two Rao-Blackwellized particle filters that jointly estimate the target trajectory and the additional state variables needed to represent the switching models. The first filter is a standard implementation using the prior importance function for the model. The second algorithm is an auxiliary particle filter that includes observation data in the resampling step. It yields an improvement in performance (especially noticeable when only a small number of particles can be used) at the expense

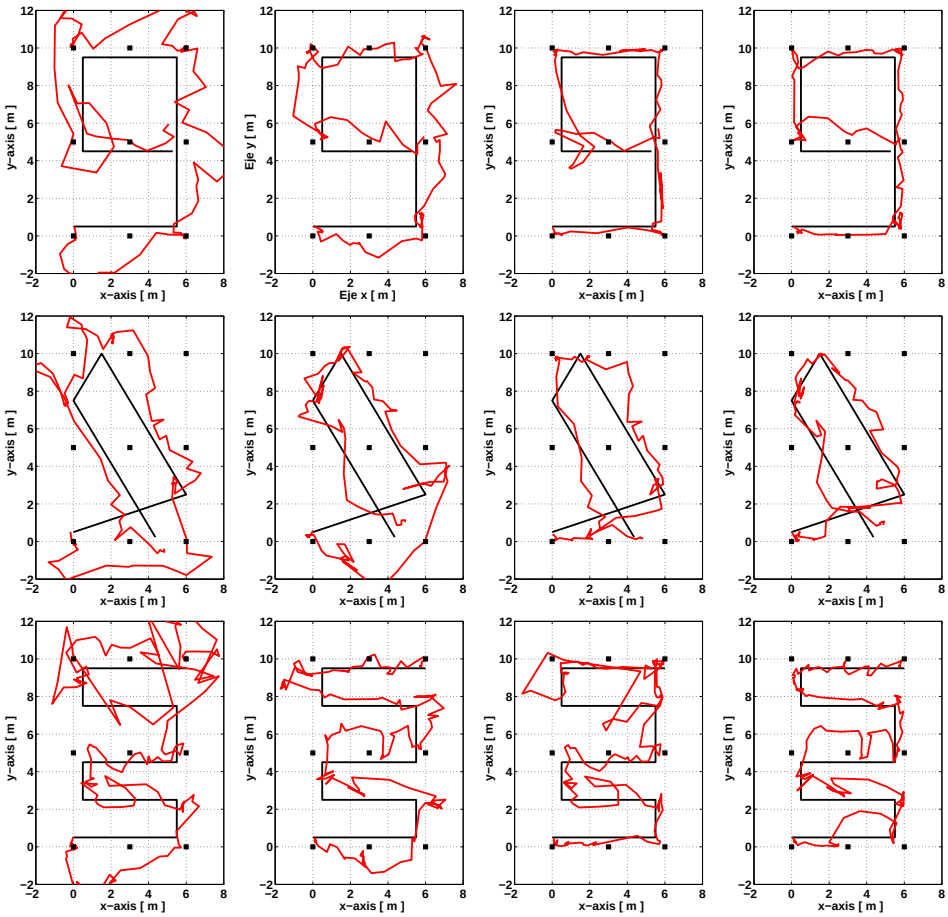


Figure 3.8: Tracking performance of the A-RBPF algorithm with experimental data: results of target tracking with three different reference trajectories and real RSS observations collected in the experimental setup described in Section 3.3.1. The real trajectories are plotted in solid black lines, the estimated trajectories in solid red lines and the sensors are depicted with dark squares. To perform tracking we have used four different algorithms that correspond to an A-RBPF scheme that uses the four different likelihood functions we introduced. In each column we show the tracking performance of a different algorithm and in each row we show the tracking capabilities of the algorithms for a specific trajectory.

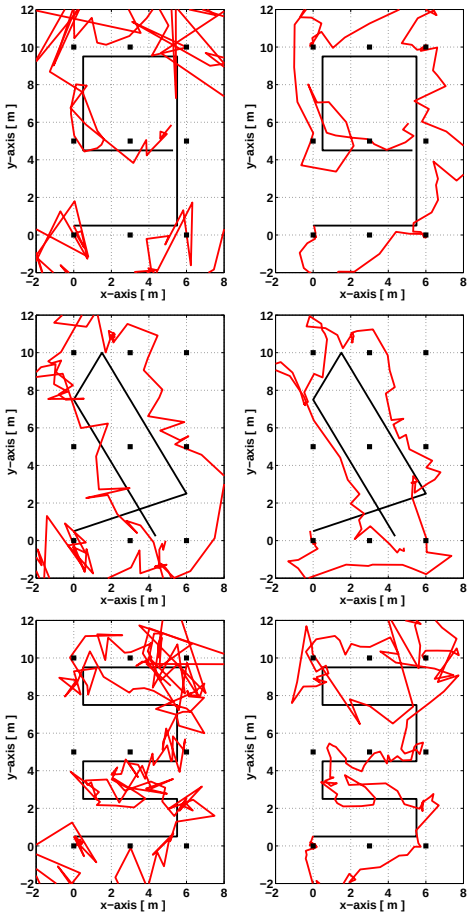


Figure 3.9: Tracking performance of the A-RBPF algorithm of 200 particles and the IMM-UKF with experimental data. The real trajectories are plotted in solid black lines, the estimated trajectories in solid red lines and the sensors are depicted with dark squares. All the algorithms use logarithmic observation models with Gaussian likelihoods. The column in the left shows the outcome of the IMM-UKF tracker and the column in the right illustrates the performance of the A-RBPF algorithm.

of little extra computational complexity.

We have provided numerical results that illustrate the performance of the proposed methods with both synthetic and experimental RSS measurements. The experimental setup to obtain the data for the assessment of the algorithms consisted of a sensor network of nine IEEE 802.15.4 sensors deployed in a 6×10 meter area. Using real data from this setup, we have constructed two sets of observation sub-models. The first set involves polynomials of high order fitted with a large amount of data. The second set consists of (simpler) logarithmic sub-models. They involve few parameters to adjust and only two sub-models for the whole network. The data for model fitting is obtained from the messages transmitted at the network startup, and hence the procedure can be made automatic. We have tried the two observation model schemes with two types of likelihoods: a Gaussian likelihood and a truncated Gaussian likelihood that incorporates a priori information about the boundaries of the area where the target can move. The numerical assessment of these two schemes shows that the polynomial sub-models yield a considerable performance advantage with respect to the logarithmic sub-models when the boundaries of the motion area are unknown. However, when the latter information is available, the simpler logarithmic models attain nearly the same performance as the polynomial ones, and hence should be preferred.

Finally, note that a further performance improvement can be achieved by constructing larger sets of observation sub-models (provided that sufficiently rich data are available). However, handling more sub-models also brings an increased computational complexity. In this paper we have restricted ourselves to relatively simple configurations in order to show that a practically meaningful performance can be achieved with a moderate computational burden.

Algorithm	M	Rnd. trajectory		Trajectory 1		Trajectory 2		Trajectory 3	
		MAE [m]	SDE [m]	MAE [m]	SDE [m]	MAE [m]	SDE [m]	MAE [m]	SDE [m]
RBPF	100	1.788	1.258	1.539	0.945	1.870	1.094	1.715	1.076
Gauss log.	500	1.603	1.121	1.412	0.863	1.736	1.016	1.584	0.986
A-RBPF	100	1.759	1.265	1.530	0.951	1.864	1.095	1.714	1.100
Gauss log.	500	1.600	1.125	1.414	0.868	1.736	1.017	1.581	0.983
RBPF trunc.	100	1.444	0.844	1.280	0.718	1.364	0.760	1.307	0.751
Gauss log.	500	1.368	0.788	1.216	0.690	1.307	0.729	1.240	0.706
A-RBPF trc.	100	1.414	0.839	1.260	0.722	1.353	0.762	1.288	0.753
Gauss log.	500	1.362	0.788	1.215	0.691	1.307	0.731	1.238	0.707

Table 3.8: Mean of 500 independent simulations for the mean absolute error of the position, in units of meters, and standard deviation of the error for the two proposed algorithms and the single logarithmic observation model for a random trajectory and for the specified 3 trajectories. The first line corresponds to the results for $M = 100$ particles, the second line for $M = 500$ particles. The first two rows display results with algorithms that use Gaussian likelihoods and the bottom two rows display the results for the algorithms that use truncated Gaussian likelihoods.

Chapter 4

A distributed particle filter implementation

We propose a mathematically sound distributed particle filter for target tracking in a real-world indoor WSN comprised of low-power nodes. We provide formal and general descriptions of the methodology and then present the results of both real-world experiments and computer simulations that use models fitted with real data. In particular, we have carried out real-world tests that show how the proposed distributed particle filter can successfully track a moving target using a network of passive light sensors.

The chapter is organized as follows. In Section 4.1 we provide a brief introduction to the methods that have been investigated in the distributed target tracking literature. Section 4.2 describes the dynamic and observation models we consider. Section 4.3 provides details of the specific deployment for the experiments and the fitting of the observation models. In Section 4.4 we provide a formal description on the DRNA algorithm. Both numerical and experimental results are presented and discussed in Section 4.5 and, finally, Section 4.6 is devoted to the conclusions.

4.1 Introduction

Distributed applications of tracking in WSN are particularly interesting in situations where high powered centralized hardware cannot be used. For example, in deployments where computational infrastructure and power are not available or where there is no time or trivial way of connecting to it, or in situations where powerful processing hardware is too large or expensive to consider for a practical deployment. In distributed target

tracking applications, signal processing tasks need to be shared by the multiple nodes of the WSN. Note that many items in the literature often refer to WSNs as being "distributed", even when processing is centralized, because they are merely referring to the physically distributed nature of WSNs. See, for example [28, 42]. In this paper we refer to "distributed" specifically with regard to processing, meaning that the computational tasks are divided among a set of low-power devices in the WSN.

Stochastic filtering methods [10] are obvious candidates for distributed tracking applications and so they have been researched by many authors in the context of WSNs [87, 84, 37, 34]. Many works use *consensus* based algorithms [68, 53], where local nodes transmit estimates to neighboring nodes, in order to, in an iterative manner, fuse the information and reach to an agreement about the global estimate on a local basis. Other popular distributed approaches are based on diffusion (i.e., distributed cooperative) techniques, [26], where local nodes exchange their estimates with neighbors in order to *improve* their estimates and fuse the collected estimates .

Distributed particle filtering for target tracking in wireless sensor networks has already attracted much attention [30, 105, 60, 55, 57]. In [16], a fully decentralized particle filtering algorithm for cooperative blind equalization is introduced. The technique is proper, in the sense that it does not make any approximations in the computation of the importance weights of the particles. However, the scheme is applicable only when the state signal is discrete, and would be infeasible in terms of computation and communication among nodes (the authors provide a simulation only) in WSNs such as we consider. In [30], the communication load is reduced using quantization and parametric approximations of densities. A similar parametric approach is applied in [56] to further simplify communications. In [15], on the other hand, methods for the parallelization of the resampling step that speed up the PF while guaranteeing that the importance weights assigned to the particles are proper are introduced [81]. Recently, a class of interacting PFs has been proposed for multi-target tracking [36, 29]. This class of algorithms relies on splitting the state-space into lower dimensional subspaces in order to become computationally tractable, but does not guarantee that the particles are assigned proper weights.

The large majority of existing contributions related to particle filtering, however, only offer a theoretical perspective or computer simulation studies, owing in part to the complications of real-world deployment and testing on low-powered hardware. Deployments of physical sensor networks have so far been purely centralized (from the computational point of view held in this paper) or, when truly distributed versions are proposed, they

are approximations to the centralized PF whose convergence cannot be guaranteed [30]. For example, [2] uses 25 acoustic sensors with a centralized PF to track a remote-controlled car, while [1] uses the RSS measurements to track an additional moving target node, also with a centralized PF.

In this work we investigate the use of the distributed resampling with non-proportional allocation (DRNA) algorithm, that was originally proposed in [15] for the parallelization of the SIS algorithm [41], for the implementation of a distributed PF running on a WSN. The DRNA was first introduced with the aim of speeding up the processing of the PF via the parallelization of the resampling step (see also [81]). It was therefore designed to be implemented on several processing elements within a single board (i.e., similar to state-of-the-art graphics processing cards [75, 109, 95]) where high-capacity communications between processing elements are guaranteed. The method is very appealing because it ensures that the importance weights of the particles are proper. Here, we tackle the problem of implementing the DRNA algorithm in a practical WSN. We first revisit the standard PF and its combination with the DRNA algorithm, providing a formal description of the methodology. This includes a short derivation showing that the DRNA procedure is unbiased.

For the practical implementation of the methodology on a real-time WSN, we have developed a software and hardware testbed with the required algorithmic and communication modules, working on a network of wireless light-intensity sensors. We assess the tracking performance of the resulting system in terms of the error obtained with both synthetic and real data. Finally, we study the constraints in the real-time operation and the communication capabilities (compared to a centralized PF) by way of experiments with our testbed implementation.

4.2 System model

4.2.1 Motion model

We consider a four dimensional state-space vector $\mathbf{x}_t = [\mathbf{r}_t^\top, \mathbf{v}_t^\top]^\top \in \mathbb{R}^4$ that describes the position $\mathbf{r}_t = [r_{1,t}, r_{2,t}]^\top$ and velocity $\mathbf{v}_t = [v_{1,t}, v_{2,t}]^\top$ of the target at time t.

Assume that the region of surveillance is a rectangle defined by two intervals, $A_1, A_2 \subset \mathbb{R}$, such that a target is in the surveillance area if its position, $\mathbf{r}_t$, fulfills $r_{1,t} \in A_1$ and $r_{2,t} \in A_2$. Then we model the target

dynamics as

$$\mathbf{x}_t = \mathbf{b}_A\big(\mathbf{f}(\mathbf{x}_{t-1}, a_t) + \mathbf{u}_t\big) \tag{4.1}$$

where

- $\{a_t\}_{t\geq 1}$ is an i.i.d. sequence of indicator random variables with probability mass function $p(a_t = 1) = \alpha_1 = 0.1$ (hence $p(a_t = 0) = 1 - \alpha_1 = 0.9$),
- $\mathbf{f}(\mathbf{x}_{t-1}, a_t)$ is a vector-valued state transition function (specified below),
- $\mathbf{u}_t$ is the process noise, with density $N(\mathbf{u}_t|\mathbf{0}, \mathbf{C}_u)$ and

$$\mathbf{C}_u = \begin{bmatrix} \sigma_r^2\mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & \sigma_v^2\mathbf{I}_2 \end{bmatrix},$$

- and $\mathbf{b}_A$ is a 'wrapper' function designed to keep the target motion within the limits of the region A (also described below).

The indicator a_t determines the kind of motion of the target. If $a_t = 0$, then $\mathbf{f}_t(\cdot, 0)$ yields a constant-velocity [11] model, namely

$$\mathbf{f}(\mathbf{x}_{t-1}, 0) = \begin{bmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{1,t-1} \\ r_{2,t-1} \\ v_{1,t-1} \\ v_{2,t-1} \end{bmatrix}$$

where T_s is the time discretization period. If, on the other hand, $a_t = 1$, then $\mathbf{f}(\cdot, 1)$ produces a sharp turn by generating a velocity vector independent of the velocity at time $t - 1$, specifically

$$\mathbf{f}(\mathbf{x}_{t-1}, 1) = \begin{bmatrix} x_{1,t-1} \\ x_{2,t-1} \\ z_t \cos(\omega_t) \\ z_t \sin(\omega_t) \end{bmatrix}$$

where z_t is the modulus of the velocity at time t, drawn from the uniform pdf $p(z_t) = \mathcal{U}(0, Z_{max})$, and ω_t is the angle of the velocity at time t, drawn from the uniform pdf $p(\omega_t) = \mathcal{U}(0, 2\pi)$. The maximum velocity is set to $Z_{max} = 1.5$ m/s.

In simulations the target will reverse any of its velocity components when moving out of the scenario bounds (intuitively, it will 'bounce' off the walls).

This reflects the fact that target motion is restricted by the bounds of an indoor scenario. In particular, recalling that $A = A_1 \times A_2$, we define the wrapper function $\mathbf{b}_A$ as

$$\mathbf{b}_A \begin{pmatrix} r_{1,t} \\ r_{2,t} \\ v_{1,t} \\ v_{2,t} \end{pmatrix} = \begin{bmatrix} r_{1,t} \\ r_{2,t} \\ v_{1,t} \times (-1)^{I(r_{1,t} \notin A_1)} \\ v_{2,t} \times (-1)^{I(r_{2,t} \notin A_2)} \end{bmatrix}$$

where the indicator function $I(r_d \notin A_d)$ returns 1 if and only if $\mathbf{r}_d$ falls outside the interval A_d, $d \in \{1, 2\}$.

4.2.2 Measurement model

We assume we receive J sensor observations coming from a WSN comprised of low-power nodes. Specifically, we assume that the sensors measure light intensity. Light sensors provide an integer value proportional to the amount of light they are receiving. We assume that light travels in straight lines and, over short distances, scattering from suspended particles and air molecules has a negligible effect on the amount of light reaching the sensor. Therefore, any object geometrically enclosed in the region between a light sensor and a light source can affect the values being read by that sensor. As light intensity fluctuates greatly when reflecting off walls, static and dynamic objects it is very difficult to model these type of observations as a sensor-target distance measurement. Instead, the observation coming from sensor j at time t, denoted $y_{j,t}$, is modeled as a binary observation giving a 1 or a 0 depending on the target position and some error probability.

We assume that the observations are conditionally independent across the different sensors, given the target position. This is a rather common assumption. Intuitively, it means that the observational noise at different sensors is independent. Hence we define the likelihood as

$$\begin{aligned} p(\mathbf{y}_t|\mathbf{r}_t) &= \prod_{j=1}^{J} p(y_{j,t}|\mathbf{r}_t) \\ &= \prod_{j=1}^{J} [p(y_{j,t} = 1|\mathbf{r}_t) I(y_{j,t} = 1) + p(y_{j,t} = 0|\mathbf{r}_t) I(y_{j,t} = 0)] \end{aligned} \quad (4.2)$$

where $\mathbf{r}_t$ is the target position, $I(.)$ is the indicator function and the factors $p(y_{j,t} = 1|\mathbf{r}_t)$ and $p(y_{j,t} = 0|\mathbf{r}_t)$ are computed as

$$p(y_{j,t} = 1|\mathbf{r}_t) = \begin{cases} 1 - \mathbf{F}^+ & \text{if } \mathbf{r}_t \in Z_j \\ \mathbf{F}^+ & \text{if } \mathbf{r}_t \notin Z_j \end{cases} \quad (4.3)$$

and

$$p(y_{j,t} = 0|\mathbf{r}_t) = \begin{cases} 1 - \mathbf{F}^- & \text{if } \mathbf{r}_t \notin Z_j \\ \mathbf{F}^- & \text{if } \mathbf{r}_t \in Z_j \ . \end{cases} \tag{4.4}$$

In the expression above, Z_j is the two-dimensional detection zone enclosed by the sensor j and the vertices of the light source and $\mathbf{F}^+$ and $\mathbf{F}^-$ are the *false positive* and *false negative* rates associated with the deployment (i.e., the probability of $y_{j,t} = 1$ when the target is outside Z_j, and the probability of $y_{j,t} = 0$ when the target is inside Z_j, respectively). Full details on the selection of the observation model parameters are given in Section 4.3.

4.3 Experimental set-up and observation models

In our deployment we use the Imote2. This mote hardware has a CPU set at 104 MHz, but it lacks native support for floating point operations which are, as a result, relatively much slower. It has 256kB of SRAM, 32MB SDRAM, and also 32MB of flash memory. It incorporates a radio transceiver with a maximum transmission rate of 31,250 Bytes/seconds. Power comes from three AAA batteries, enough to last from days to months depending on usage (with network activity causing the biggest drain, followed by CPU usage). Physically, the motes are about $6 \times 4 \times 2$ centimeters (including batteries). These motes are relatively powerful for sensor node hardware, but they are still very modest as compared with a typical desktop or laptop machine.

We use the simple light sensors that come with the Imote2's basic sensor board, which provide a simple positive integer reading between 0 and 65535 relative to the current light level (0 in complete darkness). This reading can be converted into the standard measurement of *lux*, although this is not necessary for our choice of observation model (recall from (4.2) that our chosen measurement model is binary).

Our experimental scenario is depicted in Figure 4.1. It is a room with $J = 10$ nodes (each equipped with a light sensor) enclosing an area of 3.2×6.0 meters with a single source of natural light (a window). Although there is little variation in artificial light, natural light comes more readily from side-on, making it more practical to carry out indoor tracking without modification to the existing setup of the room (by adding lamps, etc.).

Using light sensors for position estimation is fundamentally different from using other types of observations such as RSS, sound waves, GPS signals, etc., which are functions of the distance between the target and the sensor. Light sensors, on the other hand, simply provide an integer

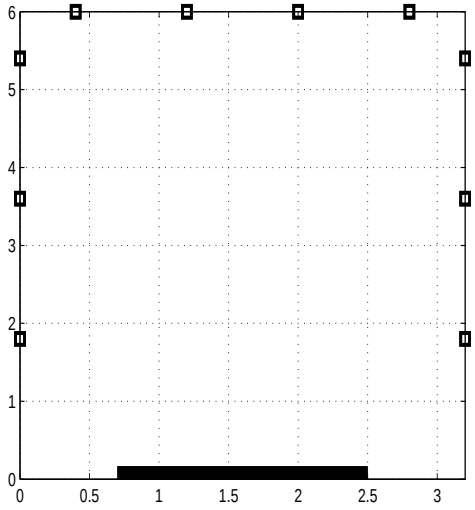


Figure 4.1: Tracking scenario of 3.2×6.0 meters (a bird's-eye view). The bold line indicates the light source (a window). There are $J = 10$ motes equipped with light sensors are arranged around the edges, indicated by squares. The entry to the scenario lies at the bottom-right corner.

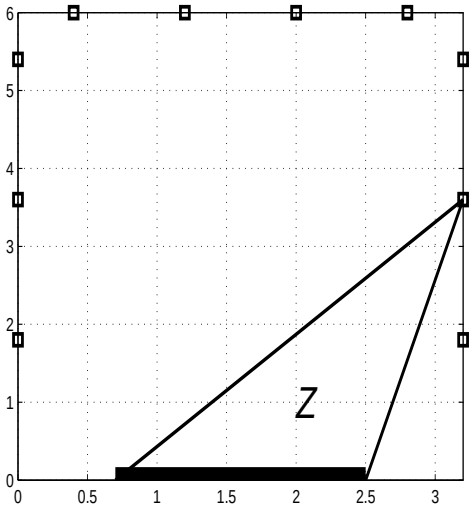


Figure 4.2: The detection zone of a sensor, labeled Z in the plot, is the area enclosed by a triangle with one vertex at the sensor and the other two vertices at the sides of the light source.

value proportional to the amount of light they are receiving. In a uniform medium (such as it is, approximately, the air inside a room), light travels in straight lines, and over short distances (e.g., several meters, as we are considering) scattering from suspended particles and air molecules has a negligible effect on the amount of light reaching the sensor. Therefore, any object geometrically enclosed in the region between a light sensor and a light source can affect the values being read by that sensor. If the source of light is a window, in a top-down two-dimensional representation the detection zone can be viewed as a triangle, as depicted in Figure 4.2 (i.e., the Z region). See [71] for details.

Reflections and scattering from surfaces (such as walls), inanimate objects (e.g., furniture) and the target itself can have a major effect on the amount of light directed towards a sensor, but are far too complex to be modeled, especially on our very limited hardware. Since it is very hard to translate the disturbances caused by the target in the sensor readings into distance measurements, we instead focus on obtaining binary observations: 1 if the target is inside the detection zone and 0 otherwise.

As the target keeps moving within the detection zone, it may block light

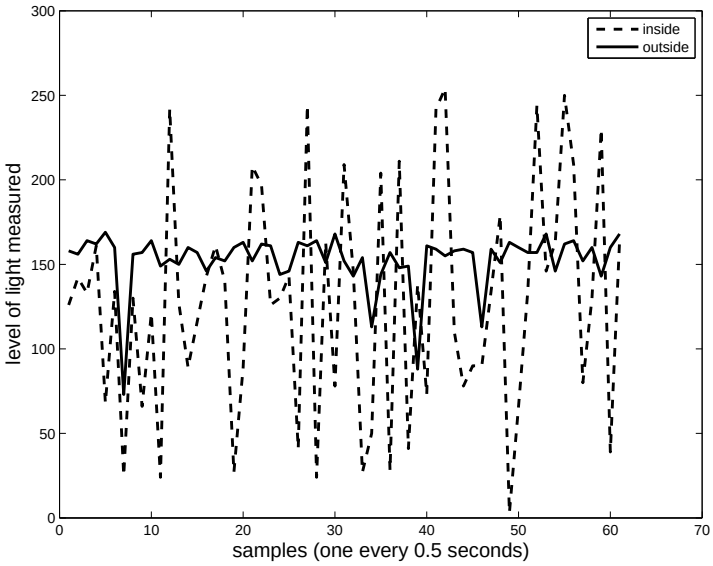


Figure 4.3: The solid line represents the light signal measured when the target (a walking person) moves randomly outside the detection zone. The dashed line is the signal when the target moves randomly inside the detection zone. Rather than a reduction in the light level, the presence of a target within the detection zone causes a large variance in the sensor readings.

or actually reflect light into the light sensor, thus causing the light level reading to go either up or down. This effect is illustrated in Figure 4.3. Furthermore, under natural light there is no base level for the measured signal (when no target is present) due to changes in the weather or the time of day. Therefore, it is very difficult to detect the target presence from the mean or instantaneous amplitude of the light signal. Instead, the short-term variance of light readings is very informative and becomes a reliable indicator of the presence of moving objects in the detection zone.

In order to turn this intuition into a quantitative model, we propose to process the light readings at every node in order to convert them into binary data. Between the sequential steps at times $t-1$ and t each sensor $j = 1, \ldots, J$ collects L light readings (evenly spaced in time) to produce a set $V_{t,j} = \{v_{1,t,j}, \ldots, v_{L,t,j}\}$ (with $L = 5$ in our experiments). For a given threshold τ, the j-th sensor outputs the binary datum

$$y_{t,j} = \begin{cases} 1, & \text{if } \mathrm{var}(V_{t,j}) > \tau \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

where $\text{var}(V_{t,j})$ is the empirical variance of the sample $V_{t,j}$. Putting the observations of all nodes together into a single $J \times 1$ vector we obtain, for every time step t, the full observation $\mathbf{y}_t = [y_{1,t}, \ldots, y_{J,t}]^\top$.

For our experiments, we calibrated the threshold τ a priori (offline) from a set of real-world data. Specifically, we instructed a person (acting as the target) to walk following an arbitrary trajectory within the monitored area during a period comprising T discrete time steps. At each time step $t = 1, ..., T$, each sensor j collected samples $V_{t,j} = \{v_{1,t,j}, \ldots, v_{L,t,j}\}, j = 1, \ldots, J$ (one set per sensor). With the data collected up to time T, we selected the threshold τ as the average of the empirical variances of all sensors, i.e.,

$$\tau = \frac{1}{JT} \sum_{j=1}^{J} \sum_{t=1}^{T} \text{var}(V_{t,j}).$$

The same threshold is then used by all nodes.

Recall from Section 4.2.2 that the likelihood is defined by Eqs. (4.3) and (4.4), where we only need to fix the error rates $\mathbf{F}^+$ and $\mathbf{F}^-$. These are calibrated empirically from a short supervised walk ($T = 20$ steps) in the real-world scenario. Specifically, we set

$$\mathbf{F}^+ = \frac{1}{JT} \sum_{j=1}^{J} \sum_{t=1}^{T} I(y_{j,t} = 0, \mathbf{r}_t \in Z_j)$$

and

$$\mathbf{F}^- = \frac{1}{JT} \sum_{j=1}^{J} \sum_{t=1}^{T} I(y_{j,t} = 1, \mathbf{r}_t \notin Z_j),$$

respectively, where each $y_{t,j}$ is given from Eq. (4.5) on the motes and $I(a, b)$ is an indicator function returning 1 if, and only if, both a and b are true, and 0 otherwise. In our experimental setup, this supervised walk was an X-shaped trajectory passing through all sensors' detection zones at least once. The error probabilities $\mathbf{F}^+$ and $\mathbf{F}^-$ are constants in the model, and identical for all PEs. The (supervised and a-priori known) walk used to calibrate $\mathbf{F}^+$, $\mathbf{F}^-$ is different from the (unsupervised and arbitrary) walk used to obtain the threshold τ.

Note that the constants in the likelihood model (i.e., τ, $\mathbf{F}^+$ and $\mathbf{F}^-$) are the same for all nodes. It is possible to select a unique threshold τ_j (and, hence, unique error rates $\mathbf{F}_j^+$ and $\mathbf{F}_j^-$) for every sensor by using supervised trajectories in and out of the detection zones $Z_1, ..., Z_J$. This is intuitively quite appealing from the point of view of trying to maximize the accuracy

of the observation model. However, we found that in real-world tests this approach leads to worse performance than a single general model for all nodes. This is probably due to the greater possibilities of introducing errors when manually aligning sensor readings with the true trajectory (necessary for a supervised walk), and from inadvertently obtaining overly 'sterile' data by choreographing the target's movements too precisely, as well as overfitting the data (the target used for training will not be the same one for testing). Moreover, manually aligning the binary sensor readings for the true trajectory is a very intensive task, impractical for many real-world deployments. In light of this, we decided to calibrate a single observation model, common to all sensors.

4.4 Distributed particle filtering

4.4.1 General structure

We look at the DRNA algorithm introduced in [15]. This algorithm was originally proposed to speed up the processing time of PFs by making them suitable for multi-processor devices endowed with high-speed communication networks. In this paper, we propose to apply a DRNA scheme to implement a distributed PF on a WSN, whose nodes can operate as processing elements (PEs). Let us remark that this framework is rather different from the one assumed in [15] or [81]. In particular, the PEs are low-powered devices that have to perform sensing, computation and radio communication tasks while running on batteries. Moreover, the schemes of [15, 81] are based on the assumption that all observations can be readily made available to all PEs in the system. Such capacity cannot be taken for granted in a WSN, where the observations are collected locally by the nodes and communications are necessarily constrained because of energy consumption. In the following we describe the method using, essentially, the notation of [81].

Assume we have N processing nodes (or PEs) in the network; each is capable of running a separate particle filtering algorithm with K particles (we ignore any non-processing nodes for now since they do not run particle filters). The total number of particles distributed over the network is $M = NK$. In particular, after the completion of a full recursive step of the distributed PF at time $t-1$, the n-th PE should hold the set $\left\{\mathbf{x}_{t-1}^{(n,k)}, w_{t-1}^{(n,k)*}, W_{t-1}^{(n)*}\right\}_{k=1,\ldots,K}$, where

- $\mathbf{x}_{t-1}^{(n,k)}$ is the k-th particle at the n-th PE,

- $w_{t-1}^{(n,k)*}$ is the corresponding unnormalized importance weight, and
- $W_{t-1}^{(n)*} = \sum_{k=1}^{K} w_{t-1}^{(n,k)*}$ is the unnormalized aggregated weight of the n-th PE.

Each PF run locally in a node involves the usual steps of drawing new samples, computing weights and resampling. In particular, resampling is carried out only locally, without interaction with the particles in other PEs. In order to avoid the degeneracy of the local sets of particles (e.g., when K is very low), the DRNA scheme includes a particle exchange step in which neighbor PEs (those connected directly, in a single jump, in the WSN) interchange subsets of their particles and unnormalized weights. This step involves the update of the aggregated weights, but each individual particle preserves its unnormalized importance weight, no matter its location in the network.

In the following, we describe the algorithm steps in detail, including the computation of state estimators, and finally provide a complete outline of the method.

4.4.2 Particle exchange

The particle set in the n-th PE is said to degenerate when its aggregated weight $W_t^{(n)*}$ becomes negligible compared to the aggregated weights of the other nodes. Note that having $W_t^{(n)*} \approx 0$ means that the particles in the n-th set hardly contribute to the approximation of the posterior probability distribution of interest, hence the computational effort invested in propagating them becomes a waste.

In order to keep the aggregated weights balanced, neighboring nodes can exchange subsets of particles and local unnormalized weights [81]. Assume that, at the beginning of the t-th time step, the n-th PE holds the weighted particles $\{\mathbf{x}_{t-1}^{(n,k)}, w_{t-1}^{(n,k)^*}\}_{k=1,...,K}$ (note the unnormalized weights). The n-th node will receive weighted particles from a certain set of PEs and transmit particles to another (possibly different) set of PEs. To be specific, let us denote

- $\mathcal{N}_n^{\mathsf{in}} \subseteq \{1, 2, ..., N\}$, the set of indices corresponding to the nodes that are expected to transmit a subset of their particles toward the n-th PE, and
- $\mathcal{N}_n^{\mathsf{out}} \subseteq \{1, 2, ..., N\}$, the set of indices corresponding to the nodes that expect to receive a subset of the particles generated at the n-th PE.

For regularity, assume that each PE transmits disjoint subsets of Q particles to each of its designated neighbors. In particular, let

$$\mathcal{M}_t^{n,s} = \left\{ \mathbf{x}_{t-1}^{(n,i_r^s)}, w_{t-1}^{(i_r^s)*} \right\}_{r=1,...,Q}$$

be the particles and weights transmitted from node n to node $s \in \mathcal{N}_n^{\mathsf{out}}$. The indices $i_1^s, ..., i_Q^s \in \{1, ..., K\}$ can be selected in any desired way (even randomly) as long as the messages $\mathcal{M}_t^{n,s}$ are disjoint, i.e., $\mathcal{M}_t^{n,s} \cap \mathcal{M}_t^{n,r} = \emptyset$ for any pair $s, r \in \mathcal{N}_n^{\mathsf{out}}$, $s \neq r$.

The information held by the n-th PE *after* the particle exchange at time t is given by $\{\tilde{\mathbf{x}}_{t-1}^{(n,k)}, \tilde{w}_{t-1}^{(n,k)^*}\}_{k=1}^K$ where

$$\left\{ \tilde{\mathbf{x}}_{t-1}^{(n,k)}, \tilde{w}_{t-1}^{(n,k)*} \right\}_{k=1}^K = \left(\underbrace{\{\mathbf{x}_{t-1}^{(n,k)}, w_{t-1}^{(n,k)*}\}_{k=1}^K}_{\text{initial}} \setminus \underbrace{\left(\cup_{s\in\mathcal{N}_n^{\mathsf{out}}} \mathcal{M}_t^{n,s}\right)}_{\text{transmitted}} \right) \cup \left(\underbrace{\cup_{s\in\mathcal{N}_n^{\mathsf{in}}} \mathcal{M}_t^{s,n}}_{\text{received}} \right), \quad (4.6)$$

and we assume that

$$\left| \cup_{s\in\mathcal{N}_n^{\mathsf{out}}} \mathcal{M}_t^{n,s} \right| = \left| \cup_{s\in\mathcal{N}_n^{\mathsf{in}}} \mathcal{M}_t^{s,n} \right|,$$

for every PE $n \in \{1, ..., N\}$, so that the number of particles per PE remains constant, $K = M/N$. The new aggregated weight for the n-th node becomes $\tilde{W}_{t-1}^{(n)^*} = \sum_{k=1}^K \tilde{w}_{t-1}^{(n,k)^*}$. Let us remark that the overall sets of particles and weights before and after the particle exchange are identical, i.e.,

$$\cup_{n=1}^N \left\{ \mathbf{x}_{t-1}^{(n,k)}, w_{t-1}^{(n,k)*} \right\}_{k=1,...,K} = \cup_{n=1}^N \left\{ \tilde{\mathbf{x}}_{t-1}^{(n,k)}, \tilde{w}_{t-1}^{(n,k)*} \right\}_{k=1,...,K},$$

while, in general, the aggregated weights are different, $W_{t-1}^{(n)*} \neq \tilde{W}_{t-1}^{(n)*}$.

4.4.3 Local processing

Immediately after the particle exchange at time t, the weighted particle set at the n-th PE is $\{\tilde{\mathbf{x}}_{t-1}^{(n,k)}, \tilde{w}_{t-1}^{(n,k)*}\}_{k=1,...,K}$. The generation of new particles, the update of the importance weights and the resampling step are taken strictly locally, without interaction among different nodes. To be specific, assume that the transition pdf of model (2.1) is used as an importance function and that the observation vector $\mathbf{y}_t$ is available at every node[1]. Then, at the n-th PE, and for $k = 1, ..., K$,

[1]The availability of the observations, which are typically collected locally in a WSN, involves communications among the nodes.

1. $\bar{\mathbf{x}}_t^{(n,k)}$ is drawn from the pdf $p(\mathbf{x}_t^{(n,k)}|\tilde{\mathbf{x}}_{t-1}^{(n,k)})$, and

2. the corresponding unnormalized weight is computed as

$$\bar{w}_t^{(n,k)*} = \tilde{w}_{t-1}^{(n,k)*} p(\mathbf{y}_t|\bar{\mathbf{x}}_t^{(n,k)}).$$

Hence the information stored by the n-th node at this point becomes $\{\bar{\mathbf{x}}_t^{(n,k)}, \bar{w}_t^{(n,k)*}\}_{k=1,...,K}$ and the aggregated weight is $W_t^{(n)*} = \sum_{k=1}^K \bar{w}_t^{(n,k)*}$.

Next, a resampling step is taken locally by each PE. Assuming a multinomial resampling algorithm[2], we assign, for $k = 1, ..., K$,

$$\mathbf{x}_t^{(n,k)} = \bar{\mathbf{x}}_t^{(n,j)}, \quad \text{with probability } \bar{w}_t^{(n,j)} \text{ and } j \in \{1, ..., K\},$$

where

$$\bar{w}_t^{(n,j)} = \frac{\bar{w}_t^{(n,j)*}}{\sum_{l=1}^K \bar{w}_t^{(n,l)*}}, \quad j = 1, ..., K,$$

are the locally normalized importance weights. After resampling, the particles at the n-th PE are equally weighted, namely $w_t^{(n,k)^*} = \frac{W_t^{(n)*}}{K}$. Trivially note that $W_t^{(n)*} = \sum_{k=1}^K \bar{w}_t^{(n,k)*} = \sum_{k=1}^K w_t^{(n,k)*}$, i.e., the resampling step keeps the aggregated weights invariant.

4.4.4 Estimation

Assume we are interested in the estimation of moments of the posterior distribution, e.g.,

$$(f, \mu_t) = \int f(\mathbf{x}_t)\mu_t(d\mathbf{x}_t),$$

where f is some function of the state vector at time t, $\mu_t(d\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{y}_{1:t})d\mathbf{x}_t$ is the filter probability measure and we introduce the shorthand (f, μ_t) to denote the integral of the function f with respect to the measure μ_t.

We can obtain local estimates of (f, μ_t) at any node. To be specific, we can build discrete random approximations of the measure μ_t as

$$\bar{\mu}_t^{n,K}(d\mathbf{x}_t) = \sum_{k=1}^K \bar{w}_t^{(n,k)} \delta_{\bar{\mathbf{x}}_t^{(n,k)}}(d\mathbf{x}_t),$$

[2]This can be substituted by any other procedure without affecting the rest of the algorithm.

before the resampling step, and

$$\mu_t^{n,K}(d\mathbf{x}_t) = \frac{1}{K}\sum_{k=1}^{K} \delta_{\mathbf{x}_t^{(n,k)}}(d\mathbf{x}_t),$$

after the resampling step, where $\bar{w}_t^{(n,k)} = \bar{w}_t^{(n,k)*}/W_t^{(n)*}$, $k = 1, ..., K$, are the locally normalized importance weights. These choices of discrete measures lead to the approximations

$$(f, \bar{\mu}_t^{n,K}) = \sum_{k=1}^{K} \bar{w}_t^{(n,k)} f(\bar{\mathbf{x}}_t^{(n,k)}) \quad \text{and} \quad (f, \mu_t^{(n,K)}) = \frac{1}{K}\sum_{k=1}^{K} f(\mathbf{x}_t^{(n,k)})$$

of the posterior expectation (f, μ_t).

Global estimates can be easily computed by a linear combination of the local estimates. If $W_t^{(n)} = W_t^{(n)*}/\sum_{i=1}^{N} W_t^{(i)*}$ is the globally normalized aggregated weight of the n-th node, then we can build the discrete random measures

$$\bar{\mu}_t^{N,K}(d\mathbf{x}_t) = \sum_{n=1}^{N} W_t^{(n)} \bar{\mu}_t^{n,K}(d\mathbf{x}_t), \quad \mu_t^{N,K}(d\mathbf{x}_t) = \sum_{n=1}^{N} W_t^{(n)} \mu_t^{n,K}(d\mathbf{x}_t),$$

using the local approximations either before or after resampling, respectively. The resulting global estimates are

$$(f, \bar{\mu}_t^{N,K}) = \sum_{n=1}^{N} W_t^{(n)} (f, \bar{\mu}_t^{n,K}), \quad (f, \mu_t^{N,K}) = \sum_{n=1}^{N} W_t^{(n)} (f, \mu_t^{n,K}). \tag{4.7}$$

The resampling operation carried out locally at the N nodes is globally unbiased in the sense defined in, e.g., [31, 38, 81]. To make this explicit, consider the sigma algebra generated by the random weights before resampling, i.e., $\mathcal{G}_t = \sigma - (\bar{w}_t^{(n,k)*}, \bar{\mathbf{x}}_t^{(n,k)}; n = 1, ..., N; k = 1, ..., K)$. Since the aggregated weights $W_t^{(n)*}$ are $\mathcal{G}_t$-measurable, for any integrable function f the conditional expectation of the estimator $(f, \mu_t^{N,K})$ given $\mathcal{G}_t$ is

$$\mathbb{E}\{(f, \mu_t^{N,K})|\mathcal{G}_t\} = \sum_{n=1}^{N} W_t^{(n)} \mathbb{E}\{(f, \mu_t^{n,K})|\mathcal{G}_t\} \tag{4.8}$$

and, since the local normalized weights $\bar{w}_t^{(n,k)}$ and particles $\bar{\mathbf{x}}_t^{(n,k)}$ are also $\mathcal{G}_t$-measurable,

$$\mathbb{E}\{(f, \mu_t^{n,K})|\mathcal{G}_t\} = \sum_{k=1}^{K} \bar{w}_t^{(n,k)} f(\bar{\mathbf{x}}_t^{(n,k)}) = (f, \bar{\mu}_t^{n,K}). \tag{4.9}$$

Substituting (4.9) into (4.8) yields

$$\mathbb{E}\{(f, \mu_t^{N,K})|\mathcal{G}_t\} = (f, \bar{\mu}_t^{N,K}),$$

i.e., the DRNA procedure is unbiased.

4.4.5 Summary

Table 4.1 summarizes the DPF algorithm investigated in this paper. Note that in order to apply this technique, we assume that *all* the observations in the vector $\mathbf{y}_t$ are available at *every* node at time t. This assumption is fairly natural in the parallel computation setup of [15] and [81], but not necessarily in the WSN framework of interest here. This issue will be specifically addressed in the subsequent sections.

In order to obtain a global estimate of (f, μ_t), each node n in the network should transmit its local estimate $(f, \mu_t^{n,K})$ and its aggregated weight $W_t^{(n)*}$ to a prescribed node (working as a fusion center) where $(f, \bar{\mu}_t^{N,K}) = \sum_{n=1}^{N} W_t^{(n)}(f, \bar{\mu}_t^{n,K})$ can be computed.

4.5 Simulations and experimental results

We illustrate the validity of our approach by applying the proposed DPF algorithm in a real-world WSN for target tracking using the binary observation model described in Section 4.3. We first describe the dynamic model for the target, then show results both for synthetic and experimental (real-world) observations. We conclude the section with a brief discussion of the advantages and disadvantages of the proposed scheme.

4.5.1 Setup

The prior distribution of the target state $p(\mathbf{x}_0)$ is (multivariate) Gaussian. In particular, $\mathbf{r}_0$ and $\mathbf{v}_0$ are a priori independent and

$$p(\mathbf{r}_0) = N(\mathbf{r}_0|\bar{\mathbf{r}}_0, \sigma_{r,0}^2\mathbf{I}_2) \quad \text{and} \quad p(\mathbf{v}_0) = N(\mathbf{v}_0|\bar{\mathbf{v}}_0, \sigma_{v,0}^2\mathbf{I}_2),$$

where $\mathbf{I}_2$ is the 2×2 identity matrix and

- the mean prior position is $\bar{\mathbf{r}}_0 = [2.5, 0.4]^\top$, i.e., the target enters through the bottom-right corner of the area of interest sketched in Figure 4.1,

Table 4.1: Distributed particle filtering (DPF) algorithm.

1. Initialization, at $t = 0$, for $n = 1, ..., N$:
   - Draw $\mathbf{x}_0^{(n,k)}$, for $k = 1, \ldots, K$, from the prior pdf $p(\mathbf{x}_0)$.
   - Assign equal weights to the samples, $w_0^{(n,k)^*} = \frac{1}{K}$ for every k, and set $W_0^{(n)*} = 1$.
   - Build the set $\{\mathbf{x}_0^{(n,k)}, w_0^{(n,k)*}, W_0^{(n)*}\}_{k=1}^K$.
2. Recursive step, for $t > 0$, start from the set $\{\mathbf{x}_{t-1}^{(n,k)}, w_{t-1}^{(n,k)*}, W_{t-1}^{(n)*}\}_{k=1}^K$. Then, for $n = 1, ..., N$:
   - **Exchange** particles with neighbor PEs in $\mathcal{N}_n^{\mathsf{in}}$ and $\mathcal{N}_n^{\mathsf{out}}$, as described in Section 4.4.2, to obtain the sets $\{\tilde{\mathbf{x}}_{t-1}^{(n,k)}, \tilde{w}_{t-1}^{(n,k)^*}\}_{k=1}^K$.
   - **Sampling:** Draw $\bar{\mathbf{x}}_t^{(n,k)}$ from $p(\mathbf{x}_t|\tilde{\mathbf{x}}_{t-1}^{(n,k)})$, for $k = 1, \ldots, K$.
   - **Weight update:** Compute $\bar{w}_t^{(n,k)*} = \tilde{w}_{t-1}^{(n,k)*} p(\mathbf{y}_t|\bar{\mathbf{x}}_t^{(n,k)})$.
   - **Estimation**: When needed, compute $(f, \bar{\mu}_t^{n,K})$ as explained in Section 4.4.4.
   - **Resampling**: Perform resampling locally to obtain the set $\{\mathbf{x}_t^{(n,k)}, w_t^{(n,k)*}, W_t^{(n)*}\}$, where $w_t^{(n,k)*} = W_t^{(n)*}/K$ for every $k = 1, ..., K$.

Variable	Symbol	Value (unit)
No. of PEs	N	$4^{\dagger}$
No. of nodes	J	10
Total no. of particles	M	100
No. of particles / PE	K	M/N
No. of exchanged particles	Q	1
No. of timesteps	T	18
Sampling Period	T_s	1.0 (seconds)
Position variance	σ_p^2	0.5 (square meters)
Velocity variance	σ_v^2	0.004 (square meters)
Maximum velocity	V_{max}	1.5 (meters/second)

† varied for some experiments

Table 4.2: DPF and model parameters.

- the mean prior velocity is $\bar{\mathbf{v}}_0 = [-0.2, 0.2]^{\top}$, i.e., the target is initially expected to move toward the middle of the area of interest, and
- the prior variances are $\sigma_r^2 = 0.5$ and $\sigma_v^2 = 4 \times 10^{-3}$.

Table 4.2 displays the values of the relevant simulation and algorithm parameters. Note that the number of PEs N is the only variable which we change directly in our experiments. For example, we use $N = 1$ as the equivalent to a centralized PF (for comparison). However, changing N affects other variables, such as the number of particles per PE ($K = M/N$). The sampling period, T_s, is set to one second. This means that each node produces one binary observation ($y_{j,t}$) per second.

Note that the number of particles transmitted by each PE in the particle exchange step is $Q = 1$. Also recall that the exchange is carried out in a circular manner. Local resampling is carried out at each step, which keeps the computational load even for all t and eliminates the overhead of checking if resampling is necessary. We use systematic resampling [24] which is significantly faster than, e.g., multinomial resampling schemes.

4.5.2 Computer simulations

In order to compare the proposed DPF scheme with a standard CPF, we generated 100 random and independent target trajectories $\mathbf{x}_{0:T}$, with associated synthetic data $\mathbf{y}_{1:T}$ drawn from the probability mass function $p(\mathbf{y}_t|\mathbf{x}_t)$ specified in Section 4.3. Then we applied both the CPF ($N = 1$) and the DPF scheme (with $N = 4$ PEs) with the same total number of

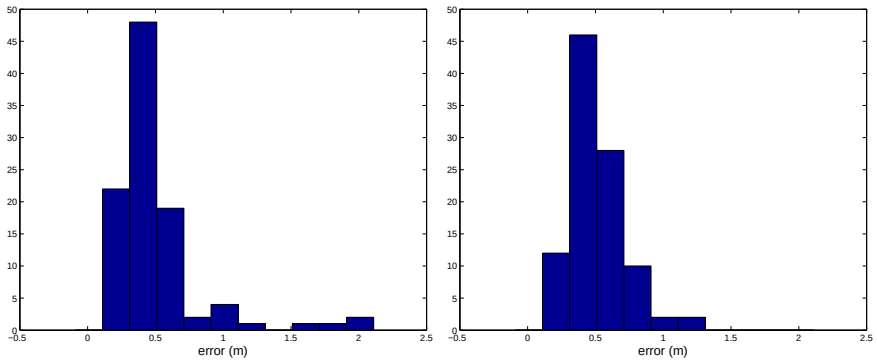


Figure 4.4: Histogram of the position error in meters for both the distributed and centralized versions of the PF (both with a total of 100 particles) over 100 simulated trajectories. The plot in the left corresponds to the results of a CPF whilst the plot in the right displays the results obtained with the DPF algorithm. In both cases the error is about half a meter on average.

particles M to track each sample trajectory from the associated sequence of synthetic observations.

Figure 4.4 displays the empirical distribution of errors, and the average error, for 100 simulated paths. For a sequence of position estimates $\hat{\mathbf{p}}_t = [\hat{x}_{1,t}, \hat{x}_{2,t}]^\top$, $t = 1, \ldots, T$, the absolute error at each step t is

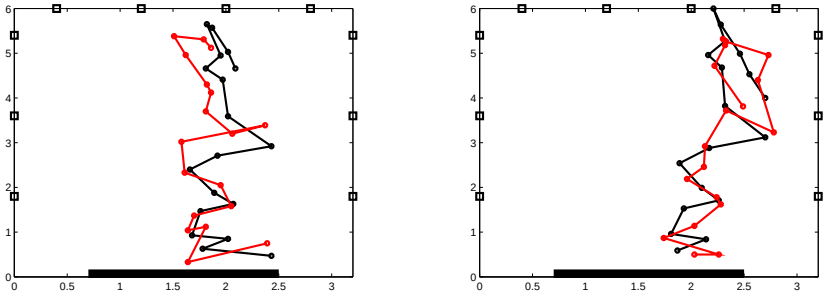
$$\epsilon_t = \sqrt{(x_{1,t} - \hat{x}_{1,t})^2 + (x_{2,t} - \hat{x}_{2,t})^2}$$

The mean absolute error (MAE) for a trajectory is

$$\bar{\epsilon} = \frac{1}{T} \sum_{t=1}^{T} \epsilon_t.$$

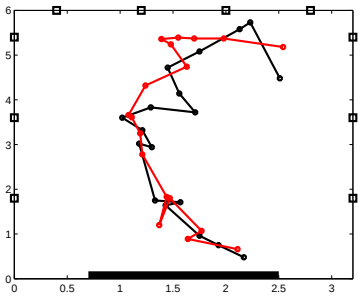
It is observed in Figure 4.4 that the performance loss in the DPF scheme is minimal.

Figure 4.5 plots a selection of these paths along with the path estimated by the DPF algorithm. Overall, there is only about half a meter of error. We also see that there is practically no difference between the performance of the CPF and the DPF algorithm. The discrepancies between true and estimated location tend to occur when the target moves between detection zones. As the observations are binary and zone-based, rather than distance-based, there are 'gaps' around the edges (see for example the final points of Run 4 in Figure 4.5). Accuracy also tends to be higher nearer the light source where more detection zones overlap (see for example, Run 1).

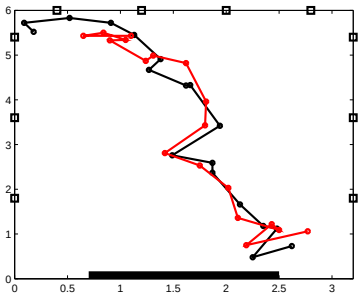


(a) Run 1

(b) Run 2



(c) Run 3



(d) Run 4

Figure 4.5: The simulated (black) paths for for simulations 1–4, and the corresponding DPF-estimated paths (red); each over $T = 18$ time steps.

4.5.3 Experimental results

Using the parameter values in Table 4.2 we carried out the following experiment to track a person walking across the monitored region.

1. The person was instructed to walk a prescribed path through our real-world scenario; specifically, between four points (dashed line in Figure 4.6) with each segment being slightly faster than the last. The trajectory lasts $T = 18$ seconds. Each node generated one observation per second, $y_{j,t}$, $j = 1, \ldots, 10$, $t = 1, \ldots, 18$.

2. As the person walked, we run the DPF algorithm with $N = 4$ PEs in *real-time* to process the vectors of observations $\mathbf{y}_t$, $t = 1, \ldots, T$, and produce filtered estimates $\hat{\mathbf{x}}_t = \sum_{n=1}^{N} W_t^{(n)} \sum_{k=1}^{K} \mathbf{x}_t^{(n,k)} \bar{w}_t^{(n,k)}$, $t = 1, \ldots, T$, of the target trajectory.

Figure 4.6 shows both the real trajectory the person was asked to walk and the path estimated by the DPF algorithm. The true path can only be drawn approximately, although we expect the path depicted in the figure to be accurate to within a fraction of a meter. Time points are unavailable, as we have no way of accurately synchronizing the target movements with the scenario, but we know the target walked each of the three segments slightly faster than the previous one and the full path took 18 seconds to complete. As we expected, the target made brief pauses of about one second at the point where the direction changed (the pauses reflected the time necessary to stop and change direction, rather than a scripted break).

The DPF implementation tracked the target quite closely. As we saw in the computer simulations, the PF can have some difficulty near the edges, where there are fewer detection zones (such as the corners), but quickly catches up when the target crosses the detection zones again. This particular trajectory in the experiment is challenging because three of the four points are right on the edge (if not slightly outside) the detection zones. At the final point, where several detection zones overlap, the target is estimated within centimeters.

4.5.4 Performance study

The speed (and also the accuracy) of the DPF scheme is influenced by a trade off between processing and network communications. We study both of these factors separately.

Table 4.3 displays the average processing time *per timestep* of DPF algorithms with $N = 1, 2, 4, 8$ ($N = 1$ corresponding to the CPF) and a

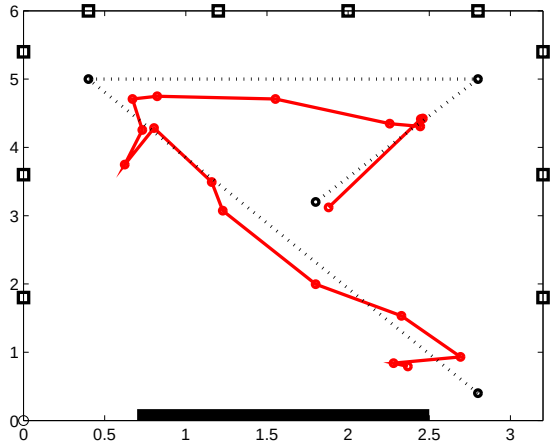


Figure 4.6: Results of the DPF algorithm (solid line, $N = 4$) tracking a target walking a prescribed trajectory (dashed line). Of the three straight lines making up the true path, each is walked slightly faster than the previous one, with a pause of about one second taken at the point where the direction is changed (indicated by hollow circles). The path is walked in $T = 18$ timesteps (18 seconds in our setup).

CPF ($N = 1$)	DPF $N = 2$	DPF $N = 4$	DPF $N = 8$
3.37	1.51	0.73	0.26

Table 4.3: The processing time (seconds) per timestep of the DPF algorithm for various values of N. The total number of particles is constant ($M = 100$); $100/N$ per PE. Note that this time does not include network activity.

constant number of particles $M = 100$ shared among all PEs ($K = M/N$ each[3]) on the earlier-described real-world trajectory. We see that halving N approximately doubles the running time. It is clear that 3.37 seconds per timestep as obtained by the CPF ($N = 1$) is not adequate for real-time tracking. Over a run of 18 seconds, this only allows for five steps through the PF. Note that a CPF simulated on an Intel Xeon 3.16GHz CPU runs at 0.0008 seconds per timestep (over 4000 times faster); a clear indication of the hardware limitations we are dealing with, as well as the efficiency of our implementation.

Table 4.4 shows how the network communication load increases with more PEs. All DPF algorithms in this layout send only one more packet per timestep than a CPF, since all nodes must broadcast their observations at each timestep (to the CPF's single PE). On the other hand, the number of bytes per timestep increases linearly with respect to the number of PEs (N), as particles must be shared among them. With $N = 4$ this equates to 90 Bytes (or 178 if needing to broadcast a global estimate) at each timestep, for which we have up to $T_s - 0.73 = 0.27$ seconds (refer to Table 4.2, and Table 4.3 for $N = 4$). As iMote2s have a maximum bandwidth of 31,250 Bytes/second, this is easily doable (even if we take into account a real-world scenario where this maximum is not achievable, packet overhead, time taken to resend dropped packets, etc., we are still left with a 'comfortable' margin).

Table 4.5 shows the main memory usage by the DPF implementation. In our model with $M = 100$ particles and $N = 4$ PEs ($K = 25$ particles per PE) we use up to 4196 Bytes per mote, a tiny fraction of the 32 Megabytes available on the iMote2. Hence, our PF implementation does not require much memory and would be suitable for deployment on motes with a lot less memory than the iMote2 (which actually has much more than is typical).

In the implementation we have presented, some optimization is still possible for faster performance. For example storing the state as integers with only 2 Bytes instead of 4, and using a single bit to store individual observations (and only broadcasting it if it is 1). This would reduce

[3]Or as close to it as possible: for $N = 8$ we round down to $K = 12$.

	CPF ($N = 1$)	DPF $N \geq 2$	DPF $N \geq 2$ with global est.
No. of Packets	$J - 1$	J	J
No. of Bytes	$J - 1$	$J + 20N$	$J + 40N + 8^{\star}$

$\star$ we are only interested in estimating the *position*-coordinates of the state (2×4 Bytes).

Table 4.4: The *network activity* (in terms of packets and bytes) per timestep for J motes comprised of N PEs and $J - N$ SEs. We a store each 4-dimensional state $\mathbf{y}_t$ with its weight w_t in 20 Bytes (4 Bytes for each number) and each observation y_t in 1 Byte.

Variable	Memory (Bytes)
Weights	$4M$
Normalized Weights	$4M$
States	$4dM$
States-buffer (used in resampling)	$4dM$
Estimations (local, global, norm. const.)	$4d + 4dN + 4N$
Layout, Observation, Constants, Misc.	100 (approx.)

Table 4.5: Memory usage for M particles, a state size of $d = 4$, and N PEs. Assuming 4 Bytes to store floating point values (as is the case on the iMote2).

network traffic (in terms of packet size) and perhaps also computational time. However, we did not wish to present an over-optimized system for a single scenario, but rather a generic one that is suitable for deployments in a variety of environments.

4.5.5 Limitations and remarks

Due to the huge range of potential applications, there are many aspects which we can not directly address. Here we remark upon some of the most important ones, that we leave for future work.

We have not considered multiple targets. Single-target is common throughout the literature, although there is nothing about our deployment that prohibits multi-target tracking algorithms.

In our deployment, if the target remains motionless it will become invisible to the DPF tracker, since the light sensors will no longer note a disturbance (i.e., the variance of light readings will stabilize). However, once the target starts moving again, the particles should quickly 'find' it.

We have not measured battery consumption directly (due to the difficulty in doing so with the specific hardware we have available). Although we can say that we have not needed to replace the batteries throughout several hours of testing, clearly the network we present could not run continuously for weeks on end, as it requires a relatively large amount of processing and radio traffic as compared to many other WSN applications. Nevertheless, it would be trivial to put the network in a sleep mode while no activity was detected by any sensors.

Our study has shown that the application of a mathematically-sound DPF scheme in a WSN is indeed feasible on very low-powered hardware, and this gives way to many potential real-world applications.

4.6 Conclusions

We have described the implementation of a distributed particle filter for target tracking in a WSN. Unlike other works in the literature, our method guarantees that the particle weights are constructed properly, and hence also the state estimators. We have carried out a series of simulations using models fitted with real light intensity data that show a tracking precision of around half a meter. In this respect, the performance difference between the proposed distributed particle filter and a centralized filter with the same total number of particles is less than two centimeters, whereas only the distributed version is fast enough for real-world deployment on the hardware

we consider. To support this claim we have implemented a real-world WSN to track a moving target in a 3.2×6.0 meter indoor scenario using only light-intensity measurements; accuracy is also to within about half a meter on average.

The distributed particle filter with four processing nodes is over four times faster than an equivalent centralized version, meaning equivalently that the same performance can be obtained on less powerful hardware. A greater proportion of processing nodes does imply more reliance on efficient communications, but applications are mainly limited only by the overall size of the network. In our network all nodes must make their observations available to all processing nodes at each time step; hence the communications load grows directly with the total number of nodes in the network, and proportionally to the dimensionality of these observations. Adaptations to this filter are needed to scale up to much larger networks with higher dimension observations; for example, only requiring observations from an active area, or working with out-of-sequence observations. We have no reason to believe that such adaptations are not possible (we partially address this issue in Chapter 5), and we have shown that a DRNA-based particle filter is already suitable to WSN scenarios, on hardware thousands of times slower than a typical desktop machine.

Chapter 5

A Distributed particle filter for wireless sensor networks with stochastic observation exchange

In this chapter we introduce a novel distributed PF for target tracking in multi-hop, possibly large, WSNs. The methodology is built around the DRNA algorithm. The DRNA technique guarantees the properness of the particle approximations produced by the filter, but it places stringent demands on the communication between nodes that make its implementation impractical for large WSNs. We investigate how to relax the communication load by using (i) a random model for the spread of data over the WSN and (ii) methods that enable the out-of-sequence processing of sensor observations.

The rest of the chapter is organized as follows. Section 5.1 describes existing distributed particle filtering methods for multi-hop WSNs and we discuss their advantages and disadvantages. In Section 5.2 we present the signal and observation models that we assume for the tracking problem. The proposed DPF is introduced in Section 5.3. Section 5.4 is devoted to the analysis of the relationship between the communication load in the WSN and the accuracy of the proposed method, as well as the handling of out-of-sequence measurements. Some illustrative computer simulations are reported in Section 5.5 and, finally, a brief discussion of the obtained results and open issues is presented in Section 5.6.

5.1 Introduction

As explained in Chapter 4, the problem of implementing PFs in a distributed fashion has drawn considerable attention in the past few years. Depending on the application, the goals of a DPF may be to speed up the the processing of data by sharing the computational load among several sites [15, 81] or to reduce the communication burden when the relevant data are available at different nodes [30, 56].

The DPF based on the DRNA algorithm guarantees the computation of proper weights and consistent estimators provided that the whole set of observations $\mathbf{y}_{1:t}$ is available at every node at time t. We have explored this approach in Chapter 4, where not only a DRNA-based PF has been derived but we have also shown experimental results of its physical implementation on a WSN of relatively small size with binary observations. Unfortunately, due to practical communication constraints, the technique, as described so far, may turn out unrealistic for many WSNs of larger size or WSNs that produce larger volumes of data.

In this chapter, we investigate how to spread the observations over the network using (i) a random model for the spread of data over the WSN and (ii) methods that enable the out-of-sequence processing of sensor observations. The transmission of data over the network is carried out using stochastic Markov chain models, akin to the scheme of [76] but we focus on spreading the observations instead of spreading all the particles and their weights over the network. A numerical illustration of the performance of the new algorithm compared to a centralized PF and the DPF based on the DRNA algorithm of Chapter 4 is also provided.

5.2 System model

We consider the problem of tracking a target that moves along a 2-dimensional region and transmits a radio signal. We assume that $N = 16$ sensors are uniformly placed in fixed positions in a 60×60 m area forming a mesh. The closest distance between two sensors is, thus, 15 meters and the furthest distance is 63.63 meters. Each sensor measures the RSS of the signal transmitted by the target.

The CV motion model and the RSS observation model adopted for this problem both were introduced in Section 2.2.1. That section, though, explained the models in a generic manner without setting any parameters. A similar observation model was also used in Section 3.3.3, however the

parameters settings for that model were adapted to an indoor scenario, which resulted in a specific characterization of the RSS measurements. Here we discuss a setup which is more suited to outdoor applications.

For clarity purposes we briefly describe again the models here and we include the values adopted for the parameters (such as the variances, the observation period, etc.).

5.2.1 Motion model

The state signal consists of the position and velocity of the target along each dimension, i.e., the 4×1 vector, $\mathbf{x}_t = [r_{1,t}, r_{2,t}, v_{1,t}, v_{2,t}]^\top \in \mathbb{R}^4$, where $[r_{1,t}, r_{2,t}]^\top$ denotes the target position and $[v_{1,t}, v_{2,t}]^\top$ denotes its velocity, both in a 2-dimensional space. The state vector has a known Gaussian prior, $p(\mathbf{x}_0) = N(\mathbf{x}_0|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$, and evolves with time according to the stochastic difference equation [50]

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{Q}\mathbf{u}_t, \tag{5.1}$$

where $\mathbf{A} = \begin{bmatrix} \mathbf{I}_2 & T_s\mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{I}_2 \end{bmatrix}$ and $\mathbf{Q} = \begin{bmatrix} \frac{1}{2}T_s^2\mathbf{I}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & T_s\mathbf{I}_{23} \end{bmatrix}$ are known, $\mathbf{I}_2$ is a 2×2 identity matrix, $\mathbf{0}_2$ is the 2×2 all-zero matrix, $T_s = 0.25$ is the observation period and $\mathbf{u}_t$ is Gaussian noise of zero mean and covariance matrix $\boldsymbol{\Sigma}_u = \begin{bmatrix} \sigma_r^2\mathbf{I}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \sigma_v^2\mathbf{I}_3 \end{bmatrix}$, with the variance parameters $\sigma_r^2 = 0.5$ and $\sigma_v^2 = 0.1$.

5.2.2 Measurement model

The measurement collected by the j-th sensor at time t is denoted as $y_{j,t}$ and its relationship with the target position, $\mathbf{r}_t$, is described by the outdoor path-loss model [98]

$$y_{j,t} = 10\log_{10}\left(\eta + \frac{P_0}{d_{j,t}^\gamma}\right) + \varepsilon_{j,t}, \tag{5.2}$$

where $P_0 = 1\text{mW}$ is the power that the target transmits; $\eta = 10^{-7}\text{mW}$ is the minimum power that an RSS sensor can measure; $\gamma = 3$ is the path loss exponent, $d_{j,t} = \|\mathbf{r}_t - \mathbf{s}_j\|$ is the distance between the position of the j-th sensor, $\mathbf{s}_j$, and the position of the target at time t, $\mathbf{r}_t$; and $\varepsilon_{j,t} \sim N(\varepsilon_{j,t}; 0, \sigma_\varepsilon^2)$ is normally distributed, zero-mean noise with known variance $\sigma_\varepsilon^2 = 2$.

5.3 Distributed tracking algorithm

5.3.1 General structure

The general structure of the proposed random spread DPF is similar to that of the DPF scheme presented in Chapter 4. We assume we have N PEs in the network, each running a separate particle filter with K particles. The total number of particles distributed over the network is therefore $M = NK$.

Each local PF performs the usual steps of drawing new samples, computing weights and resampling. After the completion of a full recursive step of the local PF at time $t-1$, the n-th PE should hold the set $\left\{\mathbf{x}_{t-1}^{(n,k)}, w_{t-1}^{(n,k)*}, W_{t-1}^{(n)*}\right\}_{k=1,\ldots,K}$, where

- $\mathbf{x}_{t-1}^{(n,k)}$ is the k-th particle at the n-th PE,
- $w_{t-1}^{(n,k)*}$ is the corresponding unnormalized importance weight, and
- $W_{t-1}^{(n)*} = \sum_{k=1}^{K} w_{t-1}^{(n,k)*}$ is the unnormalized aggregated weight of the n-th PE.

Between two local PF iterations, the DPF based on the DRNA technique of Chapter 4 would spread the *whole set* of observations along the network and then perform a particle exchange step. In a multi-hop WSN as the one we investigate here this step may easily turn out impractical unless the target WSN has a small number of nodes.

Unlike the scheme used in Chapter 4, where the observations were broadcast, here the communications are performed exclusively between neighbors. Given this constraint, we propose to spread the observations collected in the sensors over the network in a random manner, jumping from one node to another. The maximum number of allowed retransmissions is fixed a priori and it may be relatively large so that the time taken to make all the retransmissions spans over more than one PF iteration.

Sampling and resampling are performed the same way as in the standard DRNA algorithm (see Sections 4.4.1 and 4.4.3 for details) and thus these operations are not going to be described again in this chapter. The particle exchange step, in the other hand, is affected by the observation spread scheme and requires a more detailed study. As some observations may arrive to a PE at a later stage[1], the weight update step and the estimation procedure also need to be revisited.

[1]The observation $y_{j,t}$ collected in sensor j at time t may only reach sensor $i \neq j$ at a later time $t+m$ ($m > 0$).

In the following we describe the algorithm steps that are different from the DPF based on the DRNA scheme of Chapter 4: the observation spread scheme, the weight update, the particle exchange and the estimation. Then we provide a summary of the complete algorithm.

Throughout this chapter we keep on using the notation of Chapter 4 where: the * superscript in $w_t^{(n,k)*}$ denotes non-normalized weights, the bar in $\bar{\mathbf{x}}_t^{(n,k)}$ indicates that the particles have not yet gone through the resampling step, the tilde in $\tilde{\mathbf{x}}_t^{(n,k)}$ indicates that the particles at node n and time t have undergone a particle exchange step.

5.3.2 Observation spread model

The transmission of observations over the network is performed stochastically. At each time step, each observation performs L "jumps" over the network. To each observation we attach a retransmission counter that indicates how many jumps have been needed to take the observation from the node where it was originally measured to the present node. This counter has to be transmitted *together* with the measurement and it is denoted $c_{s,\tau}$ for the observation $y_{s,\tau}$, where s is the sensor index and τ the time instant when $y_{s,\tau}$ was collected.

There is an upper bound on the number of times an observation can be retransmitted, possibly over several time steps, denoted B. Therefore, the observation $y_{s,\tau}$ keeps being transmitted from one node to another (neighboring) node until $c_{s,\tau} = B$. Specifically, the observations are going to be retransmitted over B/L iterations of the PF, L times per iteration.

Intuitively, B can be seen as the parameter that tunes the precision of the filter in exchange for communication load. The higher B is set, the more likely it is for the whole set of observations to reach all of the PEs in the WSN. When all the observations have reached all the PEs, the new DPF scheme becomes the DPF scheme based on the standard DRNA technique presented in Chapter 4 and obtains its best possible precision.

As the observation spread over the network is critical for the precision and we want to fix B a priori, we provide an analysis, in Section 5.4.2, of the relation between B and the probability that an observation collected at a node reaches each one of the remaining nodes in the WSN.

The number of jumps per times step, L, on the other hand, can be seen as the parameter that controls the speed of convergence towards the precision determined by B (in practice, however, this parameter also influences the precision of the filter, as it is shown in Section 5.5.7).

In order to perform the observation exchange five control and data sets are needed in each PE:

- $\mathcal{I}_t^n$, is the set of the latest (from $t-d$ to t, where d is fixed to $d = B/L - 1$) indices of observations processed at node n and time t,
- $\mathcal{A}_t^n$, is the set of the latest (from $t-d$ to t) observations collected at node n and time t (note that these observations are collected during the observation exchange step and they may be generated at node n or any other node in the network),
- $\mathcal{B}_t^n$, is the set of observations to be processed at node n and time t,
- $\mathcal{Y}_t^n(\ell)$, is the set of observations pending retransmission at node n, at time t and "jump" ℓ, and
- $\mathcal{R}_t^n(\ell)$, is the set of observations received at node n, at time t and "jump" ℓ.

Next we describe each one of them as we describe the observation exchange process.

The set of indices of processed observations, $\mathcal{I}_t^n$

Every node needs to keep track of the latest locally processed observations, either collected by itself or received from a different node. Specifically, at time t, node n maintains a set with the indices (node and time of origin) of each observation that has been processed from $t-d$ (where d is fixed to $d = B/L - 1$) up to time t, denoted $\mathcal{I}_t^n \subseteq \{1,\ldots,N\} \times \{t-d,\ldots,t\}$. The elements of $\mathcal{I}_t^n$ are pairs of the form (s,τ) and $(s,\tau) \in \mathcal{I}_t^n$ if, and only if, the observation $y_{s,\tau}$ has been received and processed at node n, and τ lies between $t-d$ and t. Obviously, $(n,\tau) \in \mathcal{I}_t^n$ for every $\tau \in \{t-d,...,t\}$ (the observations $y_{n,t-d:t}$ obtained locally at the site of node n have been processed at node n). Note that in order to keep the amount o memory low, every pair $(s,\tau) \in \mathcal{I}_t^n$, with $\tau < t-d$ is erased from the set.

These sets, $\mathcal{I}_t^n$ for $n = 1,\ldots,N$, are needed each time a weight of a particle needs to be updated. This way we avoid incorporating a likelihood more than once into a weight. These sets are specifically used in the observation exchange step and the particle exchange step (as it is later shown in Section 5.3.4).

The set of observations pending retransmission, $\mathcal{Y}_t^n(\ell)$

The set of observations pending retransmission, $\mathcal{Y}_t^n(\ell)$, contains all the observations that are going to be transmitted from node n to its neighbors at time t and "jump" index ℓ. Note that the ℓ index here refers to the "jump" index in the retransmission step and that for $\ell = 1, \ldots, L$ the PF discrete time index t is fixed.

Before we start the observation exchange at time t, the set $\mathcal{Y}_{t-1}^n(L)$ is available. Note that $y_{s,\tau} \in \mathcal{Y}_{t-1}^n(L)$ if, and only if, $y_{s,\tau}$ was received from a neighbor node $r \in \mathcal{N}_n$ at time $t-1$ and "jump" L, and $c_{s,\tau} < B$. First, an observation is collected at the local node n and added to the local set of pending retransmission, i.e.,

$$\mathcal{Y}_t^n(0) = \mathcal{Y}_{t-1}^n(L) \cup \{(y_{n,t}, c_{n,t} = 0)\},$$

and the jump index is reset, i.e., $\ell = 0$. Then, for each observation available in node n, $y_{s,\tau} \in \mathcal{Y}_t^n(\ell)$, a neighbor $r \in \mathcal{N}_n$ is chosen randomly with a uniform probability $\frac{1}{N_n}$, where N_n is the cardinality of $\mathcal{N}_n$. Before the observation $(y_{s,\tau}, c_{s,\tau}) \in \mathcal{Y}_t^n(\ell)$ is transmitted to the neighboring node r, the "jump" counter is updated, $c_{s,\tau} = c_{s,\tau} + 1$.

The set of received observations, $\mathcal{R}_t^n(\ell)$

The union of all the received pairs of observation and counter, $(y_{s,\tau_s}, c_{s,\tau_s})$, from each of the neighbors $s \in \mathcal{N}_n$ at time t and jump ℓ, makes up the set of received observations at node n, i.e.,

$$\mathcal{R}_t^n(\ell) = \cup_{s \in \mathcal{N}_n} \{(y_{s,\tau_s}, c_{s,\tau_s})\}.$$

Then, the pair of indices, (s, τ_s) belonging to each pair of observation and counter in $\mathcal{R}_t^n(\ell)$ is compared to the set of indices of observations processed at the local node, $\mathcal{I}_t^n$.

The set of observations to process, $\mathcal{B}_t^n$

The set of observations to process, $\mathcal{B}_t^n$ contains all the observations that have reached node n at time t that were not processed before in node n. The set of observation contained in $\mathcal{B}_t^n$ are later used in the weight update step. The set is initialized at time t and jump $\ell = 0$ as $\mathcal{B}_t^n = \{(y_{n,t}, c_{n,t} = 0)\}$, then in each jump ℓ it is updated adding the new information only.

Specifically, following the comparison of $\mathcal{R}_t^n(\ell)$ and $\mathcal{I}_t^n$, if a received observation, $y_{s,\tau_s} \in \mathcal{R}_t^n(\ell)$, was not processed before, $(s, \tau_s) \notin \mathcal{I}_t^n$, it is

added to the set of observations to process in node n, i.e.,

$$\mathcal{B}_t^n = \mathcal{B}_t^n \cup \{y_{s,\tau_s} \in \mathcal{R}_t^n(\ell) : (s, \tau_s) \notin \mathcal{I}_t^n\}.$$

Then the corresponding pairs of node index and time instant are added to the set of latest processed observations, $\mathcal{I}_t^n = \mathcal{I}_t^n \cup \{(s, \tau_s)\}$.

The fact that the indices are stored as processed when they have not yet been processed may be slightly confusing, however, any observation in the set of observations to process, $\mathcal{B}_t^n$ is later used in the weight update step. Adding the indices to $\mathcal{I}_t^n$ immediately avoids repeated entries in $\mathcal{B}_t^n$ that could happen if the observations have not yet been processed but have gone through the the same node more than one time during the retransmission process.

The set of available observations, $\mathcal{A}_t^n$

Next, the retransmission counter, $c_{s,\tau}$, of each of the observations in the set of received observations, $y_{s,\tau} \in \mathcal{R}_t^n(\ell)$, is checked. If the retransmission counter of a received observation has not reached its limit, i.e., if $c_{s,\tau} < B$, then the received observation is added to the set of pending retransmissions, that is,

$$\mathcal{Y}_t^n(\ell + 1) = \{y_{s,\tau} \in \mathcal{R}_t^n(\ell) : c_{s,\tau} < B\}$$

so that the observation is available for the next "jump".

This stochastic random jump process is repeated L times and once the final L-th iteration has finished, we update the set of available observations as,

$$\mathcal{A}_t^n = \{y_{s,\tau} \in \mathcal{A}_{t-1}^n : \tau \geq t - d\} \cup \mathcal{B}_t^n.$$

The set of available observations, $\mathcal{A}_t^n$, collects *all* the *unique* observations that have gone through node n, from $t - d$ to t. This set is required for the particle exchange step but it needs to be updated in the observation exchange step.

Summary

Table 5.1 describes the observation exchange step produced at each time $t > 0$ and node n for the proposed DPF scheme. Note though, that at $t = 0$, the sets, $\mathcal{I}_t^n$, $\mathcal{A}_t^n$ and $\mathcal{Y}_t^n(L)$ need to be initialized to $\mathcal{I}_0^n = \emptyset$, $\mathcal{A}_0^n = \emptyset$ and $\mathcal{Y}_0^n(L) = \emptyset$.

Table 5.1: Random spread of the observation data for the proposed DPF scheme.

1. Initialization, at each $t > 0$, and given $\mathcal{Y}_{t-1}^n(L)$, $\mathcal{I}_{t-1}^n$ and $\mathcal{A}_{t-1}^n$:
   - Collect the new data, $y_{n,t}$, at local node n and add it to
     - the retransmission set, $\mathcal{Y}_t^n(0) = \mathcal{Y}_{t-1}^n(L) \cup \{(y_{n,t}, c_{n,t} = 0)\}$,
     - the set of observations to process, $\mathcal{B}_t^n = \{(y_{n,t}, c_{n,t} = 0)\}$.
   - Update processed observation set, $\mathcal{I}_t^n = \mathcal{I}_{t-1}^n \cup \{(n,t)\}$.
2. Stochastic jump iterations, for $\ell = 1, \ldots, L$:
   - For each observation pending retransmission at node n, $y_{s,\tau} \in \mathcal{Y}_t^n(\ell)$, choose randomly a neighbor node $r \in \mathcal{N}_n$ with uniform probabilities $\frac{1}{N_n}$.
   - Update the transmission counter $c_{s,\tau} = c_{s,\tau} + 1$ and transmit the observation $(y_{s,\tau}, c_{s,\tau})$ to node r.
   - Receive one pair $(y_{s,\tau_s}, c_{s,\tau_s})$ from each neighbor node $s \in \mathcal{N}_n$ and generate the received observation set $\mathcal{R}_t^n(\ell) = \cup_{s \in \mathcal{N}_n} \{(y_{s,\tau_s}, c_{s,\tau_s})\}$.
   - If a received observation, $y_{s,\tau_s} \in \mathcal{R}_t^n(\ell)$, was not processed before, i.e., $(s, \tau_s) \notin \mathcal{I}_t^n$,
     - add it to the set of observations to process, $\mathcal{B}_t^n = \mathcal{B}_t^n \cup \{y_{s,\tau_s} \in \mathcal{R}_t^n(\ell) : (s, \tau_s) \notin \mathcal{I}_t^n\}$, and
     - update processed observation set, $\mathcal{I}_t^n = \mathcal{I}_t^n \cup \{(s, \tau_s)\}$.
   - Let $\mathcal{Y}_t^n(\ell + 1) = \{y_{s,\tau} \in \mathcal{R}_t^n(\ell) : c_{s,\tau} < B\}$ be the new set of observations pending retransmission.
3. Let $\mathcal{A}_t^n = \{y_{s,\tau} \in \mathcal{A}_{t-1}^n : \tau \geq t - d\} \cup \mathcal{B}_t^n$ be the new of set of available observations.

5.3.3 Weight update

In the weight update step, for each observation in the set of observations to process, $y_{s,\tau} \in \mathcal{B}_t^n$, we compute the likelihood of the particles of the local node, $\{\mathbf{x}_\tau^{(n,k)}\}_{k=1}^K$, and multiply it by the weights of node n, i.e.,

$$\bar{w}_t^{(n,k)^*} \propto w_{t-1}^{(n,k)^*} p(y_{s,\tau}|\mathbf{x}_\tau^{(n,k)}), \quad k = 1, ..., K. \tag{5.3}$$

Note that the weights at time t are updated with observations collected at time τ and these two may not be equal, that is, the observations $y_{s,\tau}$ may arrive at node n out-of-sequence. This means that we need to store locally the particle trajectories, $\mathbf{x}_{t-d:t}^{(n,k)}$, from $t-d$ up to time t, in order to have the particle sample, $\mathbf{x}_\tau^{(n,k)}$, associated to time instant τ.

5.3.4 Particle exchange step

As the spread of the observation data over the network is performed in a random manner, the standard particle exchange scheme, as defined in Section 4.4.2, may not be proper. Neighboring nodes indeed have a high probability of receiving a very similar set of observations, however these sets are not necessarily identical. One consequence of this is that particles of neighboring nodes may have weights built with different sets of observations. Therefore, an exchange of particles between neighbors may lead to the nodes holding particles whose weights have been computed from different observations.

Note that if the weights of a subset of particles are built with a different set of observations the normalization procedure prior to the estimation and resampling steps becomes incorrect. Due to the difficulty of computing all the normalization constants (see Section 2.3.2 for details) the algorithm works with unnormalized weights and performs the local normalization

$$\bar{w}_t^{(n,k)} = \frac{\bar{w}_t^{(n,k)*}}{\sum_{j=1}^K \bar{w}_t^{(n,j)*}} \tag{5.4}$$

when needed. The soundness of this normalization procedure relies on the constants of each weight being the same (even if unknown to us). If the weights have been built up with different sets of observations the constants would be different and (5.4) would no longer be a correct normalization.

If we want the normalization to be proper, particle weights need to be "synchronized", that is, weights with missing likelihood information need to

be updated during the particle exchange step. In order to do so, PEs use the information provided by the set of processed observation indices $\mathcal{I}_t^n$ and the set of available observations $\mathcal{A}_t^n$.

Particle synchronization

The synchronized particle exchange is performed as follows. At time t, node n transmits its indices of processed observations $\mathcal{I}_t^n$ to all of its neighbors, $r \in \mathcal{N}_n$, and receives in exchange the sets of indices of processed observations from its neighbors, $\{\mathcal{I}_t^r; r \in \mathcal{N}_n\}$. The n-th PE then compares for every $r \in \mathcal{N}_n$ the indices of the sets $\mathcal{I}_t^r$ with its own $\mathcal{I}_t^n$.

If $\mathcal{I}_t^n \subseteq \mathcal{I}_t^r$ then neighbor r is chosen as a receiver. Let us denote $\mathcal{N}_{n,t}^{\mathsf{out}} \subseteq \{1, 2, \ldots, N\}$ as the set of indices corresponding to the nodes that, at time t, expect to receive a subset of the particles generated at the n-th PE. Following the comparison of indices, the set is built as $\mathcal{N}_{n,t}^{\mathsf{out}} = \{r \in \mathcal{N}_n; \mathcal{I}_t^n \subseteq \mathcal{I}_t^r\}$.

If on the other hand $\mathcal{I}_t^r \subseteq \mathcal{I}_t^n$ then neighbor r will not receive particles from node n, but will send particles *to* node n. Let us denote $\mathcal{N}_{n,t}^{\mathsf{in}} \subseteq \{1, 2, \ldots, N\}$ as the set of indices corresponding to the nodes that, at time t, expect to transmit a subset of their particles to the n-th PE. Then this set is built as $\mathcal{N}_{n,t}^{\mathsf{in}} = \{r \in \mathcal{N}_n; \mathcal{I}_t^r \subseteq \mathcal{I}_t^n\}$.

If $\mathcal{N}_{n,t}^{\mathsf{out}} \neq \emptyset$, node n transmits $\left|\mathcal{N}_{n,t}^{\mathsf{out}}\right|$ disjoint subsets of Q particles to each of its designated neighbors. In particular, let

$$\mathcal{M}_t^{n,r} = \left\{\bar{\mathbf{x}}_{t-d:t}^{(n,i_s^r)}, \bar{w}_{t-d:t}^{(n,i_s^r)*}\right\}_{s=1,\ldots,Q} \tag{5.5}$$

be the particles transmitted from node n to node $r \in \mathcal{N}_{n,t}^{\mathsf{out}}$. The indices $i_1^r, ..., i_Q^r \in \{1, ..., K\}$ can be selected in any desired way (even randomly) as long as the messages $\mathcal{M}_t^{n,r}$ are disjoint, i.e., $\mathcal{M}_t^{n,r} \cap \mathcal{M}_t^{n,s} = \emptyset$ for any pair $r, s \in \mathcal{N}_{n,t}^{\mathsf{out}}$, $r \neq s$. Note that instead of transmitting particle instances, $\{\bar{\mathbf{x}}_t^{(n,i_s^r)}, \bar{w}_t^{(n,i_s^r)*}\}_{s=1,\ldots,Q}$, we transmit particle trajectories, $\left\{\bar{\mathbf{x}}_{t-d:t}^{(n,i_s^r)}, \bar{w}_{t-d:t}^{(n,i_s^r)*}\right\}_{s=1,\ldots,Q}$ in order to provide the samples from $t-d$ to t to the neighbor and to be able to compute any likelihood with observations that come out-of-sequence within that time window.

If $\mathcal{N}_{n,t}^{\mathsf{in}} \neq \emptyset$, node n receives $\left|\mathcal{N}_{n,t}^{\mathsf{in}}\right|$ messages that may need synchronization. If the processed observation index set of a neighbor $r \in \mathcal{N}_{n,t}^{\mathsf{in}}$, satisfies, $\mathcal{I}_t^r \subset \mathcal{I}_t^n$, then we retrieve the observations from the set of available observation at node n, $\{y_{s,\tau_s} \in \mathcal{A}_t^n; (s, \tau_s) \notin \mathcal{I}_t^r\}$, and we

synchronize the weights as

$$\bar{w}_t^{(n,i_s^r)^*} \propto \bar{w}_t^{(n,i_s^r)^*} p(y_{s,\tau}|\bar{\mathbf{x}}_\tau^{(n,i_s^r)}), \quad i_1^r, ..., i_Q^r, \tag{5.6}$$

where the indices $i_1^r, ..., i_Q^r$ do not necessarily satisfy $i_1^r, ..., i_Q^r \in \{1, ..., K\}$ anymore. We denote the synchronized message as $\widehat{\mathcal{M}}_t^{r,n}$.

If the set of processed observations corresponding to a neighbor, $r \in \mathcal{N}_{n,t}^{\mathsf{in}}$, satisfies $\mathcal{I}_t^r = \mathcal{I}_t^n$ then the corresponding message is untouched, i.e., $\widehat{\mathcal{M}}_t^{r,n} = \mathcal{M}_t^{r,n}$.

The information held by the n-th PE *after* the synchronized particle exchange at time t is given by $\{\tilde{\mathbf{x}}_{t-d:t}^{(n,k)}, \tilde{w}_{t-d:t}^{(n,k)^*}\}_{k=1}^{P^n}$ where

$$\left\{\tilde{\mathbf{x}}_{t-d:t}^{(n,k)}, \tilde{w}_{t-d:t}^{(n,k)*}\right\}_{k=1}^{P^n} =$$
$$\left(\underbrace{\{\bar{\mathbf{x}}_{t-d:t}^{(n,k)}, \bar{w}_{t-d:t}^{(n,k)*}\}_{k=1}^{K}}_{\text{initial}} \setminus \underbrace{\left(\cup_{r\in\mathcal{N}_{n,t}^{\mathsf{out}}} \mathcal{M}_t^{n,r}\right)}_{\text{transmitted}}\right) \cup \left(\underbrace{\cup_{r\in\mathcal{N}_{n,t}^{\mathsf{in}}} \widehat{\mathcal{M}}_t^{r,n}}_{\text{received}}\right), \tag{5.7}$$

and P^n is the number of particles in node n after the particle exchange step.

As the particle exchange depends on the fulfillment of the conditions of $\mathcal{I}_t^r \subseteq \mathcal{I}_t^n$ or $\mathcal{I}_t^n \subseteq \mathcal{I}_t^r$, it may happen that exchange is performed in one direction but not in the other direction, i.e., the number of neighbors that transmit to node n may be different from the number of neighbors it receives from, $\left|\mathcal{N}_{n,t}^{\mathsf{out}}\right| \neq \left|\mathcal{N}_{n,t}^{\mathsf{in}}\right|$. Consequently, the number of particles in node n before and after the particle exchange step may be different. It could be that $P^n < K$, $P^n > K$ or $P^n = K$.

If neither of the conditions is fulfilled, i.e., if $\left|\mathcal{N}_{n,t}^{\mathsf{out}}\right| = \left|\mathcal{N}_{n,t}^{\mathsf{in}}\right| = \emptyset$ then the n-th PE would remain exactly with the same particles before and after the particle exchange step. Let us remark though that the number of particles in the whole network $M = K \times N$ before and after the particle exchange is, under all circumstances, equal.

Having a varying number of particles in a node and having a different number of particles in each node leads to various practical problems (in terms of memory and processing power allocation, etc.). This issue is solved performing the particle exchange before the resampling step. In the resampling step, particles are replicated and deleted randomly using the probabilities given by the weights however the final number of replicas desired can be fixed a priori.

Table 5.2 summarizes the synchronized particle exchange process performed by the random spread DPF scheme.

Table 5.2: Synchronized particle exchange process for the random spread DPF.

1. **Simultaneously** in the whole network:
   - Each node n transmits indices of processed observations, $\mathcal{I}_t^n$, to its neighbors $r \in \mathcal{N}_n$.
   - Each node n receives indices of processed observations, $\mathcal{I}_t^r$, from its neighbors $r \in \mathcal{N}_n$.
2. For $n = 1$ to N:
   - Compare the received indices of processed observations, $\mathcal{I}_t^r$, for $r \in \mathcal{N}_n$, with own, $\mathcal{I}_t^n$.
   - Build a new set containing indices of neighbors to transmit particles to: $\mathcal{N}_{t,n}^{\mathsf{out}} = \{r \in \mathcal{N}_n; \mathcal{I}_t^n \subseteq \mathcal{I}_t^r\}$.
   - Build a new set containing indices of neighbors to receive particles from: $\mathcal{N}_{t,n}^{\mathsf{in}} = \{r \in \mathcal{N}_n; \mathcal{I}_t^r \subseteq \mathcal{I}_t^n\}$.
   - If $\mathcal{N}_{t,n}^{\mathsf{out}} \neq \emptyset$, build messages $\mathcal{M}_t^{n,r}$ as defined in (5.5) with disjoint subsets of Q particles for each of the designated neighbors.
3. **Simultaneously** in the whole network:
   - If $\mathcal{N}_{t,n}^{\mathsf{out}} \neq \emptyset$, each node n transmits messages $\mathcal{M}_t^{n,r}$ to the neighbors that can receive information $r \in \mathcal{N}_{t,n}^{\mathsf{out}}$.
   - If $\mathcal{N}_{t,n}^{\mathsf{in}} \neq \emptyset$, each node n receives messages $\mathcal{M}_t^{r,n}$ from the neighbors that can transmit information $r \in \mathcal{N}_{t,n}^{\mathsf{in}}$.
4. For $n = 1$ to N:
   - For every received message, $\mathcal{M}_t^{r,n}$, if $\mathcal{I}_t^r \subset \mathcal{I}_t^n$ then
     - find the observations that fulfill $\{y_{s,\tau_s} \in \mathcal{A}_t^n; (s, \tau_s) \notin \mathcal{I}_t^r\}$,
     - synchronize received particle weight with (5.6) and
     - rename the synchronized message to $\widehat{\mathcal{M}}_t^{r,n}$.
   - For every received message, $\mathcal{M}_t^{r,n}$, if $\mathcal{I}_t^r = \mathcal{I}_t^n$ then set $\widehat{\mathcal{M}}_t^{r,n} = \mathcal{M}_t^{r,n}$.
   - Update the particle set according to (5.3.4) adding and subtracting received and transmitted particles.
   - Update the aggregated weight of the PE with the new weights, $\tilde{W}_t^{(n)^*} = \sum_{k=1}^{P^n} \tilde{w}_t^{(n,k)^*}$.

5.3.5 Estimation

Recall from Section 4.4.4 that we are interested in the estimation of moments of the posterior distribution, e.g.,

$$(f, \mu_t) = \int f(\mathbf{x}_t)\mu_t(d\mathbf{x}_t),$$

where f is some function of the state vector at time t, $\mu_t(d\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{y}_{1:t})d\mathbf{x}_t$ is the filter probability measure and we introduced the shorthand (f, μ_t) to denote the integral of the function f with respect to the measure μ_t.

With the described observation spread scheme, however, we can only guarantee that the observations generated at the two furthest nodes at time instant t have reached one another at time $t + d$ (where $d = B/L - 1$) with a prescribed probability (details on how to compute this probability are given in Section 5.4.2). Consequently, it can be expected that more accurate estimates can be obtained for the fixed-lag smoothing measure $\mu_{t|t-d}(dx_t) = p(x_{t-d}|y_{1:t})dx_{t-d}$. In particular, we can approximate integrals of the form

$$(f, \mu_{t|t-d}) = \int f(\mathbf{x}_{t-d})\mu_{t|t-d}(d\mathbf{x}_{t-d}),$$

where f is some function of the state vector at time $t - d$.

The discrete random approximation of the measure $\mu_{t|t-d}$ in the n-th PE is built as

$$\mu^{n,K}_{t|t-d}(d\mathbf{x}_{t-d}) = \sum_{k=1}^{K} \bar{w}_t^{(n,k)} \delta_{\mathbf{x}_{t-d}^{(n,k)}}(d\mathbf{x}_{t-d}),$$

where we use the weights at time t to approximate the smoothing measure of time instant $t-d$. With this approximation the local estimation is computed as

$$(f, \mu^{n,K}_{t|t-d}) = \sum_{k=1}^{K} \bar{w}_t^{(n,k)} f(\mathbf{x}_{t-d}^{(n,k)}).$$

Global estimates are computed as before using the aggregated weights. Specifically, defining $W_t^{(n)} = W_t^{(n)*} / \sum_{i=1}^{N} W_t^{(i)*}$ as the globally normalized aggregated weight of the n-th node, we build the discrete random measure as

$$\mu^{N,K}_{t|t-d}(d\mathbf{x}_{t-d}) = \sum_{n=1}^{N} W_t^{(n)} \mu^{n,K}_{t|t-d}(d\mathbf{x}_{t-d}),$$

using the local approximations and the resulting global estimate is

$$(f, \mu_{t|t-d}^{N,K}) = \sum_{n=1}^{N} W_t^{(n)} (f, \mu_{t|t-d}^{n,K}). \tag{5.8}$$

Note that the fusion of local estimates in order to obtain a global estimate can also be done in a hierarchical manner, that is, updating the information across the network from a local node to another, in order to avoid all nodes transmitting at the same time. Also, as the algorithm does not depend on the estimation step, the estimation can be performed offline.

5.3.6 Summary of the DPF scheme 1 (DPF-1)

Table 5.3 summarizes the proposed DPF scheme with the random spread of the observation data and the synchronized particle exchange. From now on we denote this algorithm as DPF-1.

5.3.7 Summary of the DPF scheme 2 (DPF-2)

The synchronized particle exchange process of DPF-1 has some disadvantages compared to the particle exchange step used in the DPF scheme of Chapter 4. First, it requires more processing power a) for the comparison between the sets of observations, b) in order to build the sets of neighbors to transmit to, and c) to update weights of received messages. Second, it requires more memory allocation as we need to store the set of available observations, $\mathcal{A}_t^n$, just for the particle exchange step. Third, it depends on some specific conditions being fulfilled in order for the exchange to happen and these may not be fulfilled every time instant, leading to a poor information spread. Fourth, it needs more communications as we need to transmit the sets of processed observations indices, as well as the messages.

We thus propose an alternative DPF scheme (denoted DPF-2) that performs the algorithm steps the same way as the DPF-1 except from the particle exchange step. For the DPF-2 we simply propose the particle exchange process of Section 4.4.2. This step would be performed immediately before the observation spread step.

Note, though, that with this particle exchange process there may be some distortion of the approximations induced by the weights, which are not necessarily proper anymore. Such distortion can be expected to be small, however, as shown by our computer simulations (to be shown in Section 5.5).

Table 5.3: Random spread DPF scheme (DPF-1).

1. **Initialization** at $t = 0$, for each node, $n = 1, \ldots, N$:
   - Draw $\mathbf{x}_0^{(n,k)}$, for $k = 1, \ldots, K$, from the prior pdf $p(\mathbf{x}_0)$.
   - Assign equal weights, $w_0^{(n,k)^*} = \frac{1}{K}$ for every k and set $W_0^{(n)*} = 1$.
   - Build the set $\{\mathbf{x}_0^{(n,k)}, w_0^{(n,k)^*}, W_0^{(n)*}\}_{k=1}^K$.
   - Initialise the sets: $\mathcal{I}_0^n = \emptyset$, $\mathcal{A}_0^n = \emptyset$ and $\mathcal{Y}_0^n(L) = \emptyset$.
2. **Recursive step**, for $t > 0$, and given $\{\mathbf{x}_{t-1}^{(n,k)}, w_{t-1}^{(n,k)^*}, W_{t-1}^{(n)*}\}_{k=1}^K$, for $n = 1, ..., N$, and given $\mathcal{Y}_{t-1}^n(L)$, $\mathcal{I}_{t-1}^n$ and $\mathcal{A}_{t-1}^n$:
   - Perform the **observation spread** as described in Table 5.1.
   - **Sampling:** Draw $\bar{\mathbf{x}}_t^{(n,k)}$, from $p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(n,k)})$, for $n = 1, ..., N$ and $k = 1, \ldots, K$.
   - **Weight update:** For each $y_{s,\tau} \in \mathcal{B}_t^n(L)$, update weights using Eq. (5.3).
   - **Normalization:** of weights as $\bar{w}_t^{(n,k)} = \bar{w}_t^{(n,k)^*} / \sum_{j=1}^K \bar{w}_t^{(n,j)^*}$.
   - **Estimation:** When needed, compute $(f, \mu_{t|t-d}^{n,K})$.
   - Perform the **particle exchange** as described in Table 5.2.
   - **Resampling:** Perform resampling locally to obtain the set $\{\mathbf{x}_t^{(n,k)}, w_t^{(n,k)*}, W_t^{(n)*}\}$, where $w_t^{(n,k)*} = W_t^{(n)*}/K$ for every $k = 1, ..., K$..

Table 5.4 summarizes the proposed DPF-2 scheme with the random spread of the observation data, using the alternative particle exchange scheme.

5.4 Analysis

5.4.1 Out-of-sequence measurement handling

As it was explained in Section 5.3.3 some observations may arrive at the nodes out-of-sequence, that is, at time t and node n we receive an observation related to node j taken at time instant $t - m$ where $m > 0$. Because of the assumed conditional independence of the observations, however, it is straightforward to show that the update equation of (5.3) guarantees that the weights remain proper.

Let us remember that given the assumed model of (5.2), where each observation is independent given the state, the likelihood of a particle in a PE is computed as

$$p(\mathbf{y}_t|\mathbf{x}_t^{(n,k)}) = \prod_{j=1}^{J} p(y_{j,t}|\mathbf{x}_t^{(n,k)}), \tag{5.10}$$

where J is the number of observations collected in the WSN which, in the proposed scenario, equals the number of PEs, $J = N$.

Recall also that if we use the prior importance function, we can approximate the particle weights in a sequential manner as $\bar{w}_t^{(n,k)^*} \propto w_{t-1}^{(n,k)^*} p(\mathbf{y}_t|\mathbf{x}_t^{(n,k)})$. More generally though, we can approximate the weight of a particle as

$$\bar{w}_t^{(n,k)^*} \propto w_0^{(n,k)^*} \prod_{\ell=1}^{t} p(\mathbf{y}_\ell|\mathbf{x}_\ell^{(n,k)}). \tag{5.11}$$

If we substitute (5.10) on (5.11) we then obtain

$$\bar{w}_t^{(n,k)^*} \propto w_0^{(n,k)^*} \prod_{\ell=1}^{t} \prod_{j=1}^{J} p(y_{j,\ell}|\mathbf{x}_\ell^{(n,k)}). \tag{5.12}$$

As we can see from (5.12), the weights of time instant t, are the product of the likelihoods of each time instant from 0 to t that, in turn, are the product of the likelihoods computed with each individual observation, $j = 1, \ldots, J$. Thus, any time an out-of-sequence observation reaches a node, we only need

Table 5.4: Random spread DPF scheme without synchronized particle exchange (DPF-2).

1. **Initialization** at $t = 0$, for each node, $n = 1, \ldots, N$:
   - Draw $\mathbf{x}_0^{(n,k)}$, for $k = 1, \ldots, K$, from the prior pdf $p(\mathbf{x}_0)$.
   - Assign equal weights, $w_0^{(n,k)^*} = \frac{1}{K}$ for every k and set $W_0^{(n)*} = 1$.
   - Build the set $\{\mathbf{x}_0^{(n,k)}, w_0^{(n,k)^*}, W_0^{(n)*}\}_{k=1}^K$.
   - Initialise the sets: $\mathcal{I}_0^n = \emptyset$, and $\mathcal{Y}_0^n(L) = \emptyset$.
2. **Recursive step**, for $t > 0$, and given $\{\mathbf{x}_{t-1}^{(n,k)}, w_{t-1}^{(n,k)^*}, W_{t-1}^{(n)*}\}_{k=1}^K$, for $n = 1, ..., N$ and given $\mathcal{Y}_{t-1}^n(L)$ and $\mathcal{I}_{t-1}^n$:
   - Perform the **particle exchange** as described in Section 4.4.2.
   - Perform the **observation spread** as described in Table 5.1.
   - **Sampling:** Draw $\bar{\mathbf{x}}_t^{(n,k)}$, from $p(\mathbf{x}_t|\tilde{\mathbf{x}}_{t-1}^{(n,k)})$, for $n = 1, ..., N$ and $k = 1, \ldots, K$.
   - **Weight update:** For each $y_{s,\tau} \in \mathcal{B}_t^n(L)$, update weights as

$$\bar{w}_t^{(n,k)^*} \propto \tilde{w}_{t-1}^{(n,k)^*} p(y_{s,\tau}|\bar{\mathbf{x}}_\tau^{(n,k)}), \quad \text{for} \quad k = 1, \ldots, K. \tag{5.9}$$

   - **Normalization:** of weights as $\bar{w}_t^{(n,k)} = \bar{w}_t^{(n,k)^*} / \sum_{j=1}^K \bar{w}_t^{(n,j)^*}$.
   - **Estimation:** When needed, compute $(f, \mu_{t|t-d}^{n,K})$.
   - **Resampling:** Perform resampling locally to obtain the set $\{\mathbf{x}_t^{(n,k)}, w_t^{(n,k)*}, W_t^{(n)*}\}$, where $w_t^{(n,k)*} = W_t^{(n)*}/K$ for every $k = 1, ..., K$..

to compute its likelihood and multiply it by the local weights. The downfall of this approach is that we perform local resampling steps using weights that have been built with a vector of observations that may be incomplete. As resampling reduces the diversity of the particle sample, we may loose some particle trajectories that would have survived with weights built with the complete set of observations. However, the approach is proper in the sense that, as long as we compute the likelihood of the missing observations and multiply it by the weights, even if at a later stage, the weights of the surviving particles are computed correctly.

Note that the out-of-sequence problem we have described here is different from that termed in literature as out-of-sequence measurements (OOsM) [91, 78, 90, 89]. In the OOsM case, not only has the observation not been processed but also the whole time instant, $t - m$, has not been processed at all by the filter, i.e., there has not been any sampling, weight update or resampling of particles, therefore the challenge resides in obtaining particle samples for the missing time instant in order to compute the weights with the newly arrived information. In our scenario, at time instant, $t - m$, we have sampled particles, we have computed the associated weights and we have resampled the particles, however the vector of observations used for the weight update is incomplete.

5.4.2 Propagation of observation data

Consider a Markov chain X_t, $t = 0, 1, \ldots$, taking values in the discrete space $\{1, , J\}$ with conditional probabilities given by the $J \times J$ transition matrix $\mathbf{P}$. Using the classical theory of Markov chains [100], it is easy to compute the probability that the chain moves from one state to another in n transitions. Specifically, the probability of moving from state i to state j is given by the entry in the i-th row and the j-th column of matrix $\mathbf{P}$, which we denote here as P_{ij}. In order to compute the probability of moving from state i to state j in n jumps we simply compute the n-th power of the matrix, and then we select the i-th row and the j-th column entry, i.e., $(\mathbf{P}^n)_{ij}$ [100].

On the other hand, in graph theory, the adjacency matrix $\mathbf{A}$ of a graph is defined as the matrix with entries $A_{ij} = 1$ if, and only if, the i-th and j-th nodes are connected, while $A_{ij} = 0$ otherwise. We can also define the connectivity matrix $\mathbf{C}$ of a graph as the normalized adjacency matrix, i.e. $C_{ij} \neq 0$ if, and only if, the i-th and j-th nodes are connected and $\sum_{j=0}^{n} C_{ij} = 1$. Therefore, C_{ij} can be interpreted as the probability of moving from node i to node j.

If we relate the connectivity matrix of a graph or network, $\mathbf{C}$, with the

transition matrix of a Markov chain, $\mathbf{P}$, and if we identify an observation generated at a node, i, as a Markov chain starting in a state i, we can numerically compute the probability of *reaching* node or state j in n jumps.

Specifically, it is possible to compute the probability of moving from node i to node j in exactly n jumps. In particular, this is

$$(\mathbf{P}^n)_{ij} = \sum_{k=1}^{n} (\mathbf{P}^{n-k})_{ij} f_{i,j}^k, \quad \text{for} \quad i \neq j, \tag{5.13}$$

where $f_{i,j}^n$ is the probability of reaching node j from node i in n steps *for the first time*.

If we rearrange the equation and try for different values of n,

$$\begin{aligned} f_{i,j}^1 &= (\mathbf{P}^1)_{ij}, \quad \text{for} \quad n = 1, \\ f_{i,j}^2 &= (\mathbf{P}^2)_{ij} - (\mathbf{P})_{jj} f_{i,j}^1, \quad \text{for} \quad n = 2, \\ f_{i,j}^3 &= (\mathbf{P}^3)_{ij} - (\mathbf{P}^2)_{jj} f_{i,j}^1 - (\mathbf{P}^1)_{jj} f_{i,j}^2, \quad \text{for} \quad n = 3, \end{aligned}$$

we find that for an arbitrary n, the probability of reaching node j from node i in n steps *for the first time* can be computed as

$$f_{i,j}^n = (\mathbf{P}^n)_{ij} - (\mathbf{P}^{n-1})_{jj} f_{i,j}^1 - (\mathbf{P}^{n-2})_{jj} f_{i,j}^2 - \ldots - (\mathbf{P}^1)_{jj} f_{i,j}^{n-1}.$$

Finally, if we add the probabilities of reaching node j from node i for the first time in n or less jumps, we can compute the probability reaching node j from node i in at most n specific jumps,

$$F_{i,j}^n = \sum_{k=1}^{n} f_{i,j}^k. \tag{5.14}$$

With equation (5.14) we can compute the probability of, starting from a specific node, reaching each of the nodes in the network in a specific number of jumps. A more practical measure of the connectivity of a node with respect to the rest of the nodes of a WSN may be to find out, when starting in node i which node in the WSN has the lowest probability of being visited in no more than n jumps, in other words

$$m_i^n = \text{argmin}_{k \in \{1,2,\ldots,N\}} F_{i,k}^n. \tag{5.15}$$

Furthermore, a more general idea of the whole WSN may be given by the pair

$$(i_{min}^n, j_{min}^n) = \text{argmin}_{i,j \in \{1,2,\ldots,N\}} F_{i,j}^n, \tag{5.16}$$

which is the pair of nodes, i and j, for which the probability of reaching j from i in no more than n jumps is the smallest one over the whole set of nodes in the network. Intuitively, these nodes should be far apart in the network and we should observe a relation between low probability and high distance.

In order to better understand this metric, let us define a distance, $d_n(i,j)$, that is related to the probability that an observation starting at node i reaches node j in at most n jumps. Specifically, for three different nodes, i, j, and k, $d_n(i,j) > d_n(i,k)$ if, and only if, $F^n_{i,j} < F^n_{i,k}$. Note that, as these distances depend on the probabilities, $F^n_{i,k}$, and the probabilities, in turn, depend on the number of jumps, n, the relationship between these distances may vary with the number of total jumps, n.

When selecting the number of total jumps, B, that an observation makes in the WSN (introduced in Section 5.3.2), we propose to proceed as follows

1. determine the network topology and its connectivity matrix, $\mathbf{C}$,
2. set $\mathbf{P} = \mathbf{C}$,
3. compute $F^n_{i,j}$, using (5.14), for $i,j \in \{1,2,\ldots,N\}$, where N is the number of nodes in the network, and for $n = 1,2,3,\ldots,Z$, where Z is the number of maximum jumps for which the probabilities are computed,
4. find the pair of nodes, (i^n_{min}, j^n_{min}), using (5.16), for $n = 1,2,3,\ldots,Z$,
5. select a specific jump, $n = n_{tot}$, associated with the minimum desired probability, $F^{n_{tot}}_{i,j}$, for $(i^{n_{tot}}_{min}, j^{n_{tot}}_{min})$, and
6. set the number of maximum jumps for the observations to $B = n_{tot}$.

The maximum number of jumps, n_{tot}, is equal to the number of maximum jumps for the observations, B, introduced in Section 5.3.2. The choice $B = n_{tot}$ ensures that an observation collected at an arbitrary node i of the WSN will be available at any other node j after B retransmissions with probability greater than, or at least equal to, $F^{n_{tot}}_{i,j}$. For a numerical example please see 5.5.2.

5.5 Simulation results

In this section we present an example of a target tracking problem using the system model of Section 5.2. We then perform a series of simulations to analyze the performance of the algorithm for the selected tracking example.

5.5.1 Target prior parameters and example

We assume that the prior density of the state variable is Gaussian, $p(\mathbf{x}_0) = N(\mathbf{x}_0; \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$, where the mean vector is $\boldsymbol{\mu}_0 = [0, 0, 0.1, 0.1]^\top$ and the covariance matrix is $\boldsymbol{\Sigma}_0 = diag(0.5, 0.5, 0.1, 0.1)$. With a period of $T_s = 0.25$s we have simulated 3000 runs of 200 discrete time steps each using (5.1) and we have then generated the associated synthetic observations according to (5.2).

Figure 5.1 shows one of the target trajectory runs. The figure on the left shows the 60×60 meter surveillance area, the sensor positions (depicted with squares) and the target trajectory. The beginning of the trajectory is the centre of the surveillance area heading north-east. The target then performs a turn and heads south-west passing near the sensors colored in yellow and blue. The figure on the right shows the sensor readings associated with that particular trajectory were the colors of the observations match the colors of the sensors positions in the left plot. As the observations are related to distance we can observe that the two highest peaks are colored in yellow and blue.

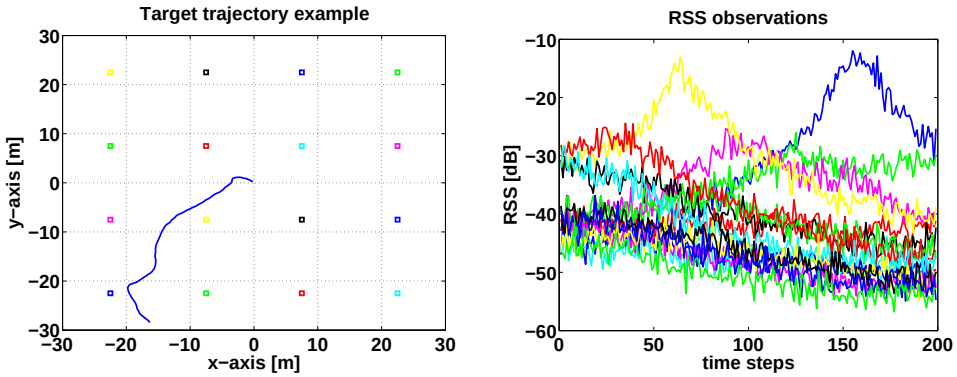


Figure 5.1: Example of two dimensional target trajectory and associated observations. The plot in the left shows the surveillance area with the sensors (depicted in squares) and the target trajectory. The plot in the right shows the sensor readings associated with that particular trajectory, where the colors of the observations match the colors of the sensor positions in the left.

5.5.2 Network connectivity and Markov chain parameter selection

Figure 5.2 left illustrates the topology of the network. The squares in the area indicate the positions of the sensors (which for this example coincide with the PEs) whilst the dashed lines indicate the connections between PEs. We have also included the PE indices for future reference. As we see the sensors are connected in such a way that the links of the whole WSN form the shape of a two dimensional square mesh. Thus, the sensors in the four corners are connected to 2 neighbors, the nodes in the sides are connected to 3 neighbors and the nodes in the centre of the grid are connected to 4 neighbors. We have assumed this topology for simplicity purposes, although the proposed DPF schemes would work in any topology. Note also that the topology affects the speed at which the information is spread over the network (however this aspect is not explored in the current work).

The links between nodes specify the set of neighbors of each node, $\mathcal{N}_n$ for each $n = 1, \ldots, N$, needed in the observation exchange step described in Section 5.3.2 and the particle exchange step described in Section 5.3.4. They also determine the transition matrix $\mathbf{P}$ of the Markov chains of Section 5.4.2. We assume the probability of transmission to each neighbor to be uniform, therefore the connectivity matrix of the network, $\mathbf{C}$, has non zero entries $C_{n,j} = \frac{1}{N_n}$ if the n-th and j-th node are connected, where N_n is the number of neighbors of the n-th node.

Using the methodology of Section 5.4.2, and using the connectivity matrix, $\mathbf{C}$, described above, we have numerically computed the probability that an observation generated at node i reaches each one of the rest of the nodes of the network for a range of values of B.

Figure 5.2 (right) shows for each of the $N = 16$ possible nodes of origin (depicted in different line colors and types) the probability of reaching the *furthest* node of the network. Note that the furthest node is the node with the highest distance, $d^B_{i,j}$, as defined in Section 5.4.2 (i.e. the node with lowest probability, $F^B_{i,j}$). Also note that more than one node can be at the furthest distance if their probability is the lowest.

If we consider the worst case scenario (magenta line of the right plot of Figure 5.2) we can select the minimum amount of jumps required to obtain a specific probability. For example, for a probability of $\pi = \min_{i,j} F^B_{i,j} = 0.8$ we require $B = 68$ jumps and for a probability of $\pi = \min_{i,j} F^B_{i,j} = 0.99$ the minimum number of jumps is $B = 180$.

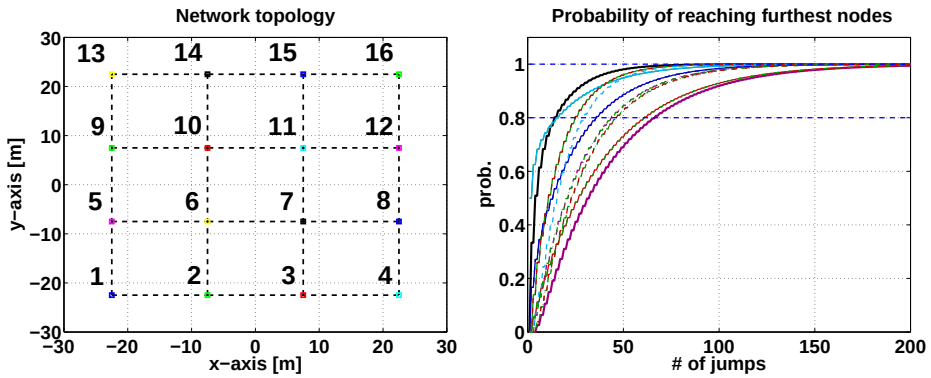


Figure 5.2: Network topology and observation spread probability.

5.5.3 Effect of the number of processing elements

In order to analyze the performance of the distributed structure provided by the DRNA scheme (i.e. the performance when all the observations are available at every node) we have performed some simulations with the CPF and the DRNA algorithm alone.

Table 5.5.3 displays the average mean absolute error (MAE) and standard deviation of the error (SDE) for the position estimates obtained with the CPF and the DRNA algorithm for different numbers of particles. Note that we compare the performance of a CPF that has $M = 3200, 8000$ and 16000 particles with a DRNA scheme that has $K = 200, 500$ and 1000 particles per PE, and thus, we are comparing filters with the same number of particles in the *whole* network.

We have selected the number of particles to be exchanged to be 5, that is, each PE transmits $Q = 5$ particles to *each* of its neighbors at each discrete time step, where the connections between the PEs (and thus the neighbors) are defined by the topology (for these experiments, a mesh). Note also that when the number of particles is incremented the computational cost increases, however we keep the number of transmitted particles fixed, $Q = 5$, and consequently the communication cost in the particle exchange step remains the same.

As expected, the best error performance is attained by the CPF, yet the difference between the two algorithms is very small (a maximum of 0.12% increased mean and 0.06% increased standard deviation on the position error) and thus we can conclude that the effect of dividing the processing among several nodes is negligible, when all the observations are present in

all nodes at all times and the number of exchanged particles is $Q = 5$.

Algorithm	# particles	MAE [m]	SDE [m]
CPF	3200	0.4043	0.2266
DRNA	200/PE	0.4044	0.2268
CPF	8000	0.4023	0.2256
DRNA	500/PE	0.4028	0.2257
CPF	16000	0.4017	0.2253
DRNA	1000/PE	0.4018	0.2254

Table 5.5: Average mean absolute error (MAE) in meters and standard deviation of the absolute error (SDE), also in meters, of the CPF and the DRNA scheme for different numbers of particles.

5.5.4 Synchronized versus non-synchronized particle exchange

In order to compare the performance between the two proposed distributed schemes (the DPF-1 and the DPF-2) we have performed simulations with a varying number of total jumps B. For these experiments we have fixed the number of intermediate jumps to the number of total jumps, that is, $L = B$ for all the different values of B.

Figure 5.3 shows the average MAE of the position in meters obtained for both algorithms when setting the total number of jumps to $B = 68$, $B = 94$ and $B = 180$, which correspond to probabilities of $\pi = 0.8$, $\pi = 0.9$ and $\pi = 0.99$. The number of exchanged particles is again $Q = 5$. Note, though, that the DPF-2 performs a particle exchange step every time instant whilst the DPF-1 performs the exchange whenever the differences between the sets of observations allow it (see Section 5.3.4). Results show that for all the different number of jumps tried, the DPF-2 algorithm attains lower error results than the filter with synchronized particle exchange (DPF-1).

As explained in Section 5.3.4, the distributed scheme of DPF-1 can only perform particle exchange steps between nodes if some specific conditions are fulfilled. Specifically, node n can only send particles to node r at time t if the condition $\mathcal{I}_t^n \subseteq \mathcal{I}_t^r$ is fulfilled. In order to analyze how frequently this condition is fulfilled we have computed the mean number of PEs that transmit particles to al least one of their neighbors.

Figure 5.4 shows the total number of PEs that perform in average at

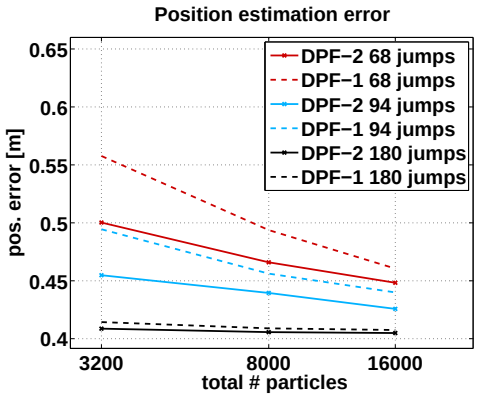


Figure 5.3: Average mean absolute error of the position in meters of the DPF-1 and the DPF-2 for different number of total jumps, B.

least one particle exchange with a neighbor per time instant when using the DPF-1 scheme. We have fixed the number of intermediate jumps to $L = 30$ and we have plotted the results for a varying number of total jumps $B = 30, 60, 90$ and 120. The number of particles used for these experiments is of $K = 200$ per PE. As we can observe, the number of processors that perform the particle exchange is relatively low (maximum of 9 out of 16 for $B = 30$) and further, it reduces as the number of retransmissions increase.

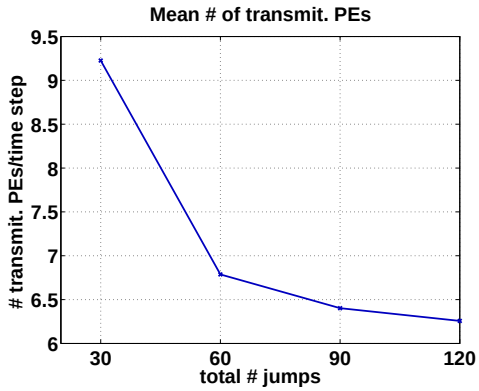


Figure 5.4: Average transmitting PEs per time instant in the particle exchange step for the DPF-1. The number of total jumps selected are $B = 30, 60, 90$ and 120 with a fixed number of intermediate jumps of $L = 30$.

Our intuition, originally, was that neighboring PEs would have similar

sets of observations, even more so if the observations are retransmitted, and consequently that the number of PEs transmitting would increase as the number of retransmissions increased. This intuition turns out in contradiction with numerical results, though. In order to understand this behavior we have, thus, analyzed the differences between sets of collected observations, $\mathcal{A}_t^n$, (as defined in Section 5.3.2) between PE $n = 7$ and its neighbors $\mathcal{N}_n = 6, 8, 13, 11$. We have selected sensor $n = 7$ because it is positioned in the middle of the network and has 4 neighbors (see Figure 5.2 left).

Figure 5.5 (left) shows the average symmetric difference between the sets of collected observations of PE $n = 7$ and the sets of its neighbors $\mathcal{N}_n = 6, 8, 13, 11$. The symmetric difference has been computed for one simulation run and has been averaged over the length of the simulation in order to compute the difference per time instant, i.e.,

$$\frac{1}{T}\sum_{t=1}^{T} |(\mathcal{A}_t^n \setminus \mathcal{A}_t^r) \cup (\mathcal{A}_t^r \setminus \mathcal{A}_t^n)|$$

where $\mathcal{A}_t^n$, is the set of observations of n-th PE at time instant t, $\mathcal{A}_t^r$ is the set of observations of the r-th PE at time instant t, $|\mathcal{A}_t^n|$ is the number of elements in the set $\mathcal{A}_t^n$, and T is the length of the simulation.

Figure 5.5 (right), in the other hand, shows the normalized symmetric difference. This is computed using the ratio of the symmetric difference and the total number of observations of both sets, i.e.,

$$\frac{1}{T}\sum_{t=1}^{T} \frac{|(\mathcal{A}_t^n \setminus \mathcal{A}_t^r) \cup (\mathcal{A}_t^r \setminus \mathcal{A}_t^n)|}{|(\mathcal{A}_t^n \cup \mathcal{A}_t r)|}.$$

As we can observe, indeed the percentage of observations that are different reduces as the observations are retransmitted (Figure 5.5 (right)) however the absolute numbers increase slightly (Figure 5.5 (left)).

The issue is that even if the sets of observations between neighbors are very similar they are hardly ever identical. When there is one difference between the sets, the exchange can happen only in one way (from one processor to another), further if there is one difference in each of the sets the particle exchange step cannot happen at all. We believe that the fact that the particle exchange does not happen very often in the DPF-1 is the main reason for its worse performance with respect to the DPF-2 filter.

In the following sections we focus on the results of the DPF-2 filter as not only it attains better error results but also it is more practical as it

does not require the memory or the processing power needed for the particle synchronization.

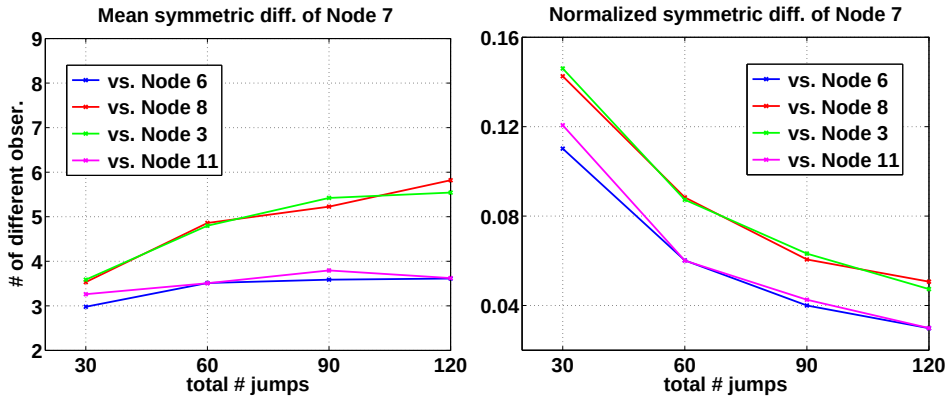


Figure 5.5: Symmetric difference and normalized symmetric difference between the sets of observations of sensor $n = 7$ and its neighbors.

5.5.5 Effect of the number of total jumps

In order to see the relation between the total number of jumps, B, and the error performance we have computed a series of simulations with the DPF-2 filter for a varying number of total jumps.

Table 5.5.5 displays the average mean absolute error (MAE) obtained for the position using the DPF-2 algorithm for a varying number of total particles, M. We have selected $B = 68, 94, 120, 180$ total jumps because they correspond to the probabilities $\pi = 0.8, 0.9, 0.95, 0.99$ that the observation coming from one of the least connected nodes has reached its furthest node in the network. For these experiments the parameter of the intermediate jumps equals the number of total jumps, i.e., $L = B$. The table also includes the percentage increase in the mean and deviation error (labelled $\%iMAE$ and $\%iSDE$) with respect to the DRNA.

As we observe, there is indeed a correspondence between the total number of jumps and the error performance. The percentage of error induced with the number of jumps is also influenced by the number of particles, for example, with $K = 200$ particles per PE and $B = 68$ jumps (thus a probability $\pi = 0.8$ for the observation spread) the mean error augments in 23% with respect to the DRNA whilst using $K = 1000$ particles the mean error increase is of 11%.

part./PE	jumps	prob. π	MAE [m]	% iMAE	SDE [m]	%iSDE
200	68	0.8	0.5002	23.68	0.2864	26.38
	94	0.9	0.4548	12.44	0.2584	14.04
	120	0.95	0.4303	6.39	0.2433	7.37
	180	0.99	0.4087	1.04	0.2293	1.17
500	68	0.8	0.4659	15.67	0.2643	17.11
	94	0.9	0.4395	9.11	0.2480	9.89
	120	0.95	0.4214	4.61	0.2374	5.20
	180	0.99	0.4057	0.72	0.2280	1.03
1000	68	0.8	0.4483	11.56	0.2534	12.43
	94	0.9	0.4258	5.96	0.2396	6.31
	120	0.95	0.4164	3.62	0.2340	3.81
	180	0.99	0.4049	0.77	0.2272	0.81

Table 5.6: Average mean absolute error (MAE) and standard deviation of the absolute error (SDE) of the DPF-2 algorithm for different types of jumps. The %iMAE and %iSDE indicate the percentage increase in the mean and standard deviation of the errors *with respect to* the DRNA.

Also it is observed the percentage error increase is quite similar for both the mean and the standard deviation.

5.5.6 Smoothed estimates

In order to analyze the improvement obtained with the observation retransmission scheme (where we select the number of intermediate jumps to be less than the number of total jumps, i.e., $L < B$) we have analyzed the error we obtain when we compute smoothed estimates. Before analyzing the retransmission scheme, though, we have computed smoothed estimates of the simulated trajectories where we have set $L = B$, that is, where there are no observation retransmissions. With these experiments we seek to find what is the improvement obtained with the smooth estimation alone.

Figure 5.6 displays the average MAE obtained with smoothed estimates of $(f, \mu_{t|t-k})$, for a varying lag $k = 0, 1, 2, \ldots, 20$, from the three algorithms. These experiments have been performed for a total number of $M = 3200, 8000$ particles for the CPF and $K = 200, 500$ for the DRNA and the DPF-2. The number of total jumps of the DPF-2 has been set to $B = 180$ (corresponding to a probability $\pi = 0.99$) and the number of intermediate jumps to $L = B$.

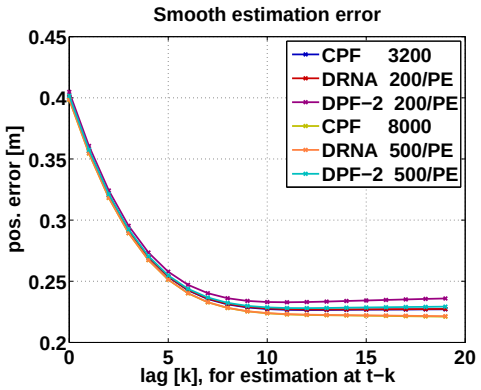


Figure 5.6: Average MAE of the position for the CPF, DRNA and DPF-2 obtained with smoothed estimates. The estimation has been performed at estimation instants of $t-k$ where $k = 0, 1, 2, ..., 20$ with observation up to t.

The three algorithms reduce their error considerably (nearly to a half of the original error) as the delay of the smoothed estimate is increased. This reduction in the error reaches a threshold which is obtained approximately at the smoothed estimate of $(f, \mu_{t|t-10})$ and then the obtained errors flatten.

5.5.7 Effect of the number of intermediate jumps

The total number of jumps B is associated with the observation spread over the network, and thus determines the precision, whilst the number of intermediate number of jumps, L, was originally associated with the speed of convergence towards the associated precision. In order to see the influence of this parameter we have computed a series of simulations where we have fixed the number of total jumps, B, and we have performed simulations for a varying number of intermediate jumps, L.

Recall, that when we fix B and L we can only guarantee with a probability π that the observations generated at time t coming from the two furthest nodes have reached one another at $t+k$, where the lag, k, is $k = B/L-1$. This means that when we want to compare the results obtained between algorithms whose intermediate jumps are different from the number of total jumps, we have to compute smoothed estimates with different lags. For example, when $L = B/3$ we are going to estimate $(f, \mu_{t|t-2})$, and when $L = B/4$, we are going to estimate $(f, \mu_{t|t-3})$.

Table 5.7 and Table 5.8 display the MAE obtained for the position using the DPF-2 algorithm for a varying number of particles, $M = 200, 500$ and

1000. For these experiments we have fixed the total number of jumps to $B \approx 68$ for Table 5.7 and to $B = 120$ for Table 5.8, which correspond to the probabilities $\pi = 0.8$ and $\pi = 0.95$ respectively. In each of the tables we show the results obtained for a varying number of intermediate jumps, $L \approx B/4, B/3, B/2, B$. Note that the smoothed estimates for the retransmission scheme of $L \neq B$ are computed for (f, μ_{t-k}), where $k = B/L - 1$. In order to make this explicit we have added an extra column indicating the lag associated with the smoothed estimate.

particles per PE	jumps B/L	lag [k] (f, μ_{t-k})	MAE	% iMAE	SDE	% iSDE
200	68/17	3	0.4871	-1.51	0.2938	5.19
	69/23	2	0.4839	-2.16	0.2850	2.04
	68/34	1	0.4867	-1.59	0.2823	1.07
	68/68	0	0.4946	N/A	0.2793	N/A
500	68/17	3	0.4192	-8.92	0.2447	-5.00
	69/23	2	0.4242	-7.84	0.2450	-4.89
	68/34	1	0.4411	-4.17	0.2511	-2.52
	68/68	0	0.4603	N/A	0.2576	N/A
1000	68/17	3	0.3868	-12.82	0.2244	-9.55
	69/23	2	0.3994	-9.98	0.2292	-7.61
	68/34	1	0.4187	-5.63	0.2355	-5.07
	68/68	0	0.4437	N/A	0.2481	N/A

Table 5.7: Average mean absolute error (MAE) and standard deviation of the absolute error (SDE) of the DPF-2 algorithm for a total of $B \approx 68$ jumps and a range of intermediate jumps $L \approx B/4, B/3, B/2, B$. The %iMAE and %iSDAE indicate the percentage increase in the mean and standard deviation of the errors *with respect to* $L = B$, that is, when the total jumps are performed in one time instant.

As we observe the results obtained for the algorithm with no retransmissions, i.e. when $B = L$, and when we transmit the total number of jumps in several iterations, i.e. when $L < B$, are very similar. For example, the results obtained for the scheme, $B = 68$ and $L = 17$ and $K = 200$ particles/PE are just %1.5 percent different to those obtained for $B = L = 68$. Similarly, the results obtained for the scheme, $B = 120$ and $L = 30$ for $K = 200$ particles/PE are %3.78 percent different from those obtained for $B = L = 120$. The results are not only very similar but slightly better for the schemes with the retransmissions (thus the *negative* increased

particles per PE	jumps B/L	lag [k] (f, μ_{t-k})	MAE	% iMAE	SDE	% iSDE
200	120/30	3	0.4097	-3.78	0.2404	1.00
	120/40	2	0.4088	-3.99	0.2360	-0.84
	120/60	1	0.4142	-2.72	0.2347	-1.38
	120/120	0	0.4258	N/A	0.2380	N/A
500	120/30	3	0.3686	-11.64	0.2124	-8.56
	120/40	2	0.3791	-9.13	0.2175	-6.37
	120/60	1	0.3929	-5.82	0.2220	-4.43
	120/120	0	0.4172	N/A	0.2323	N/A
1000	120/30	3	0.3509	-14.83	0.2030	-11.27
	120/40	2	0.3651	-11.38	0.2087	-8.78
	120/60	1	0.3828	-7.08	0.2160	-5.59
	120/120	0	0.4120	N/A	0.2288	N/A

Table 5.8: Average mean absolute error (MAE) and standard deviation of the absolute error (SDE) of the DPF-2 algorithm for a total of $B = 120$ jumps and a range of intermediate jumps $L = B/4, B/3, B/2, B$. The %iMAE and %iSDAE indicate the percentage increase in the mean and standard deviation of the errors *with respect to* $L = B$, that is, when the total jumps are performed in one time instant.

percentage). The reason for this is the fact the we use smoothed estimates. As we saw in Figure 5.6 even without retransmissions we do improve the estimation just by computing smoothed estimates. All in all, this means that schemes where the number of total jumps are performed in several iterations are comparable in terms of estimation performance when we use smoothed estimates.

Figure 5.7 displays the average MAE obtained with smoothed estimates from the DPF-2 with a fixed number of total retransmissions $B \approx 68$ (left) and $B = 180$ (right). The intermediate number of retransmissions has been set to $L \approx B/4, B/3, B/2, B$. These experiments have been performed for a total number of $K = 200$ particles per PE.

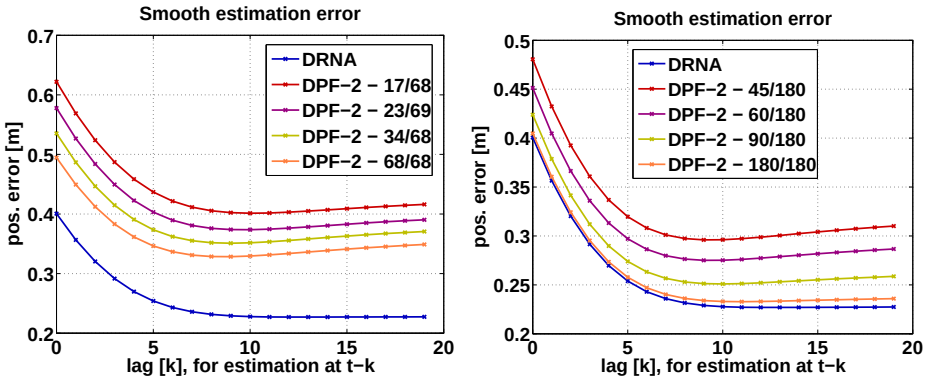


Figure 5.7: Average MAE of the position for the DPF-2 obtained with smoothed estimates. These experiments have been performed for a fixed number of total retransmissions of $B = 68$ (left) and $B = 180$ (right). We have used $K = 100$ particles per PE.

As expected, results show that the algorithms with the retransmissions in several steps, i.e., the algorithms with, $L < B$, perform worse than the algorithms whose total jumps are performed in one iteration, $L = B$. Note however that smoothed estimates of the DPF-2 scheme are similar to instantaneous estimates of the DRNA scheme. In fact, the performance of the DPF-2 scheme with $B = L = 180$ number of total jumps (and intermediate jumps) is very close to the DRNA scheme.

5.5.8 Limitations and remarks

The results show that the two proposed distributed algorithms, the DPF-2 and DPF-1, perform close to the CPF and the DRNA scheme when the

number of retransmissions for the observations is sufficiently large.

Note that the proposed retransmission scheme, where the number of intermediate jumps, L, is less than the number of total jumps, B, does not reduce the communication load with respect to a setting were $B = L$. As N new observations are generated every time instant, which need to be retransmitted up to B times, the communication bottleneck in the network ends up being $N \times B$, no matter what L is set to. The advantage of setting $L < B$ resides in reducing the algorithm processing times. As we wait for the observations to arrive the algorithm can carry on performing its sampling, weight update and resampling steps.

The cost of speeding up the processing, when setting $L < B$, is payed with the loss of precision in the instantaneous estimates. However, if smoothed estimates are used the results of performing the number of total jumps in several iterations are very similar to those obtained when all the retransmissions are performed in one iteration.

DPF-1 algorithm, which performs a synchronized particle exchange, exhibits a higher error than the DPF-2. The main reason for this is that the particle exchange step most of the time cannot happen due to the differences between the sets of observations of neighboring nodes. The issue is that even if the sets of observations between neighbors are very similar they are hardly ever identical. When there is one difference between the sets, the exchange can happen only in one way (from one processor to another), further if there is one difference in each of the sets the particle exchange step cannot happen at all. We believe that the fact that the particle exchange does not happen very often in the DPF-1 is the main reason for its worse performance with respect to the DPF-2 filter. Intuitively, not exchanging particles means reducing the number of particles of the total filter, where, in the worst case scenario, a particle filter of M particles can transform into a particle filter of $K = M/16$.

With the proposed retransmission scheme where the number of total jumps that an observation makes takes several algorithm iterations, it is necessary to compute smoothed estimates. The downfall of smoothed estimates is that it is required to store particle trajectories (and thus requires more memory storage per particle) whose length is linked to the smoothed lag or delay. The error decrease obtained, however, allows for a reduction in the number of particles as compensation.

For example, if the number of retransmissions is set to $B = L = 60$, a DPF-2 with $K = 100$ particles per PE with a smoothed estimate of $\hat{\mathbf{x}}_{t-2}$ obtains the same error as the DPF-2 with $K = 1000$ particles per PE and instantaneous estimation $\hat{\mathbf{x}}_t$. And if the filter parameters make

it more accurate, for example, if the number of retransmissions is set to $B = L = 90$, the error obtained with a DPF-2 with $K = 100$ particles per PE with a smoothed estimate of $\hat{\mathbf{x}}_{t-1}$, is comparable to that of a DPF-2 with instantaneous estimate but with $K = 1000$ particles per PE.

Even if the memory storage of the smoothed estimate of $\hat{\mathbf{x}}_{t-2}$ with $K = 100$ particles would be of 3×100, that would still be far less than the memory requirements of a filter of $K = 1000$. Further, the processing required for the sampling, weight update or resampling would still be of $K = 100$.

This characteristic improvement when using smoothed estimates is present in all filters (CPF, DRNA scheme, DPF-2 and DPF-1), even more so if the filters are more accurate, therefore the improvement of the CPF or the DRNA scheme will always be better than the improvement obtained in the DPF-2. However, the error obtained with the smoothed estimates given by the DPF-2 scheme can be compared to that of the DRNA scheme in many cases, so one could have the flexibility of the communication scheme of the DPF-2 with the accuracy of the DRNA.

5.6 Conclusions

We have introduced a distributed particle filtering scheme for target tracking in multi-hop wireless sensor networks. The proposed algorithm is built around the DRNA algorithm of [15, 81], but modifying the communication requirements. We have presented a random model for the spread of data over the WSN where the transmission of data over the network is carried out using stochastic Markov chain models. The presented observation spread scheme is flexible and allows tuning of the observation spread over the network via the selection of a parameter. We have also presented a methodology for the selection of the parameter that controls the observation spread that only requires the network connectivity information and, thus, can be performed off-line without performing any experiments. Within this scheme two parameters can be selected, the number of total retransmissions, B, and the number of intermediate retransmissions, L.

Within this random observation scheme we have presented two algorithms, one that requires a synchronization procedure of the particle weights during the particle exchange (DPF-1) and a more simple algorithm that ignores the synchronization requirements during the particle exchange step (DPF-2). The performance of the resulting distributed particle filters have been illustrated by way of computer simulations for a target tracking

problem in two dimensions in a network of 60×60 meters.

The results show that the two proposed distributed algorithms, the DPF-1 and DPF-2, perform close to the CPF and the DRNA scheme when the number of retransmissions for the observations is sufficiently large. The DPF-1 algorithm, which performs a synchronized particle exchange, exhibits a higher error than the DPF-2 scheme. The main reason for this has been shown to be that the particle exchange step cannot happen in many time instants due to the differences between the sets of observations of the neighboring nodes. The DPF-2 filter, in the other hand, has shown to attain good results even though the computation of the weights may not be fully correct for some particles.

The proposed retransmission scheme, where the number of intermediate jumps, L, is less than the number of total jumps, B, allows reducing the algorithm processing times. As we wait for the observations to arrive the algorithm can carry on performing its sampling, weight update and resampling steps. With the proposed retransmission scheme where the number of total jumps that an observations makes takes several algorithm iterations, it is necessary to compute smoothed estimates. The downfall of smoothed estimates is that it is required to store particle trajectories (and thus requires more memory storage per particle) whose length is linked to the smoothed lag or delay. The error decrease obtained, however, allows for a reduction in the number of particles as compensation. Also, if smoothed estimates are used the results of performing the number of total jumps in several iterations are very similar to those obtained when all the retransmissions are performed in one iteration.

This characteristic improvement when using smoothed estimates is present in all filters (CPF, DRNA scheme, DPF-2 and DPF-1), even more so if the filters are more accurate, therefore the improvement of the CPF or the DRNA scheme will always be better than the improvement obtained in the DPF-2. However, the error obtained with the smoothed estimates given by the DPF-2 scheme can be compared to that of the DRNA scheme in many cases, so one could have the flexibility of the communication scheme of the DPF-2 with the accuracy of the DRNA scheme.

Chapter 6

Summary and Conclusions

6.1 Summary

The aim of this work has been the design, implementation and assessment of efficient particle filters (PFs) for various specific tracking applications on wireless sensor networks (WSNs).

6.1.1 Indoor tracking with RSS measurements

The first part of the work has been focused on developing efficient models and particle filters for indoor tracking using received signal strength (RSS) in WSNs. Specifically, we have proposed a generalized switching multiple-model (GSMM) approach to the representation of the target dynamics and the radio signal-strength (RSS) observations in an indoor scenario. The resulting class of state-space models is very flexible and we claim that it may enable the adequate formal representation of time-varying scenarios with highly unstable RSS measurements. The drawback of the GSMM scheme is the increase in the dimension of the system state and, hence, the number of variables that the tracking algorithm has to estimate. To handle this difficulty, we have introduced two Rao-Blackwellized particle filters that jointly estimate the target trajectory and the additional state variables needed to represent the switching models. The first filter is a standard implementation using the prior importance function for the model. The second algorithm is an auxiliary particle filter that includes observation information in the resampling step. It yields an improvement in performance (especially noticeable when only a small number of particles can be used) at the expense of little extra computational complexity.

We have provided numerical results that illustrate the performance of the

proposed methods with both synthetic and experimental RSS measurements. The experimental setup to obtain the data for the assessment of the algorithms consisted of a sensor network of nine IEEE 802.15.4 sensors deployed in a 6×10 meter area. Using real data from this setup, we have constructed two sets of observation sub-models. The first set involves polynomials of high order fitted with a large amount of data. The second set consists of (simpler) logarithmic sub-models. They involve few parameters to adjust and only two sub-models for the whole network. The data for model fitting is obtained from the messages transmitted at the network startup, and hence the procedure can be made automatic. We have evaluated the two observation model schemes with two types of likelihoods: a Gaussian likelihood and a truncated Gaussian likelihood that incorporates a priori information about the boundaries of the area where the target can move. The numerical assessment of these two schemes shows that the polynomial sub-models yield a considerable performance advantage with respect to the logarithmic sub-models when the boundaries of the motion area are unknown. However, when the latter information is available, the simpler logarithmic models attain nearly the same performance as the polynomial ones, and hence should be preferred.

Even though some of the following articles have been references in the thesis it has to be noted that early works on Chapter 3 lead to the publishing of the following conference articles,

- K. Achutegui, L. Martino, J. Rodas, Carlos J. Escudero and J. Míguez. "A Multi-Model Particle Filtering algorithm for indoor tracking of mobile terminals using RSS data." In Control Applications,(CCA) & Intelligent Control,(ISIC), 2009 IEEE, pp. 1702-1707. IEEE, 2009.

- K. Achutegui, J. Rodas, Carlos J. Escudero and J. Míguez. "A model-switching sequential Monte Carlo algorithm for indoor tracking with experimental RSS data." In Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on, pp. 1-8. IEEE, 2010.

Finally, the following journal article can be regarded as a summary of Chapter 3

- K. Achutegui, J. Rodas, Carlos J. Escudero and J. Míguez. "A multi-model sequential Monte Carlo methodology for indoor tracking: Algorithms and experimental results." Signal Processing 92, no. 11 (2012): 2594-2613.

6.1.2 A distributed particle filter implementation on a WSN

As the main drawback of the particle filters is their potentially high computational complexity we have then focused on reducing their computational complexity via de distribution of their processing over the nodes that comprise the WSN. We have investigated the use of the distributed resampling with non-proportional allocation (DRNA) algorithm, for the implementation of a distributed PF running on a WSN. We first revisit the standard PF and its combination with the DRNA algorithm, providing a formal description of the methodology. This includes a short derivation showing that the DRNA procedure is unbiased.

We have carried out a series of simulations using models fitted with real light intensity data that show a tracking precision of around half a meter. In this respect, the performance difference between the proposed distributed particle filter and a centralized filter with the same total number of particles is less than two centimeters, whereas only the distributed version is fast enough for real-world deployment on the hardware we consider. To uphold this claim we have implemented a real-world WSN to track a moving target in a 3.2×6.0 meter indoor scenario using only light-intensity measurements; accuracy is also to within about half a meter on average.

The distributed particle filter with four processing nodes is over four times faster than an equivalent centralized version, meaning equivalently that the same performance can be obtained on less powerful hardware. A greater number of processing nodes does imply more reliance on efficient communications, but applications are mainly limited only by the overall size of the network. In our framework, network time and processing time can be traded off with each other by varying the number of nodes; but the network bottle-neck is the handling of observational data. In our network all nodes must make their observations available to all processing nodes at each time step; hence, the communications load grows directly with the total number of nodes in the network, and proportionally to the dimensionality of these observations. For the specific application we have implemented, though, where the light intensity readings are locally transformed into binary data indicating whether a target is detected by the sensor or not, the communication traffic over the WSN can be managed efficiently even with simple hardware.

It has to be noted that the work presented in Chapter 4 has been submitted to the following journal

- J. Read, K. Achutegui and J. Míguez, . "A Distributed Particle Filter for Nonlinear Tracking in Wireless Sensor Networks." submitted to

6.1.3 Distributed particle filtering on a WSN with random spread of the measurement data

The DPF based on the DRNA algorithm guarantees the computation of proper weights and consistent estimators provided that the whole set of observations is available at every node and time instant. Unfortunately, due to practical communication constraints, the technique may turn out unrealistic for many WSNs of larger size. We have therefore investigated how to relax the communication requirements using (i) a random model for the spread of data over the WSN and (ii) methods that enable the out-of-sequence processing of sensor observations.

The presented observation spread scheme is flexible and allows tuning of the observation spread over the network via the selection of two parameters. As the observation spread has a direct connection with the precision on the estimation, we have also introduced a methodology for the selection of the associated parameters. This methodology only requires the network connectivity information and, thus, can be used off-line without performing any experiments. Within this scheme two parameters can be selected, the number of total retransmissions, B, and the number of intermediate retransmissions, L, for each item of locally-collected observations.

Within this random observation scheme we have investigated two algorithms, one that requires a synchronization procedure of the particle weights during the particle exchange stage (named DPF-1) and a more simple algorithm that ignores the synchronization requirements (named DPF-2). The performance of the resulting distributed particle filters has been illustrated by way of computer simulations for a target tracking problem in two dimensions, for a WSN covering an area of 60×60 meters in an outdoor scenario.

The results show that the two proposed distributed algorithms, the DPF-1 and DPF-2, perform close to the centralized particle filter (CPF) and the standard DRNA scheme when the number of retransmissions for the observations is sufficiently large. The DPF-1 algorithm, which performs a synchronized particle exchange, exhibits a higher error than the DPF-2 scheme. The main reason for this has been shown to be that the particle exchange step cannot be carried out in many time instants due to the differences between the sets of observations of the neighboring nodes. The DPF-2 algorithm, on the other hand, has shown to attain good results even though the computation of the weights may not be proper for some particles.

The proposed retransmission scheme, where the number of intermediate jumps, L, is less than the number of total jumps, B, allows reducing the algorithm processing times. As we wait for the observations to arrive the algorithm can carry on performing its sampling, weight update and resampling steps. The downfall of selecting $L < B$, is that it is necessary to compute smoothed estimates. Smoothed estimates require to store particle trajectories (and thus require more memory storage per particle) and the length of the particle trajectory is linked to the smoothed lag or delay. The error decrease obtained, however, allows for a reduction in the number of particles as compensation. Also, if smoothed estimates are used the results of performing the number of total jumps in several iterations are very similar to those obtained when all the retransmissions are performed in one iteration.

This characteristic improvement when using smoothed estimates is present in all filters (CPF, DRNA scheme, DPF-2 and DPF-1), even more so if the filters are more accurate, therefore the improvement of the CPF or the DRNA scheme will always be better than the improvement obtained in the DPF-2. However, the error obtained with the smoothed estimates given by the DPF-2 scheme can be compared to that of the DRNA scheme in many cases, so one could have the flexibility of the communication scheme of the DPF-2 with the accuracy of the standard DRNA scheme.

Early works on Chapter 5 were published in the conference article

- K. Achutegui and J. Míguez. "A parallel resampling scheme and its application to distributed particle filtering in wireless networks." In Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2011 4th IEEE International Workshop on, pp. 81-84. IEEE, 2011.

Whilst a summary of Chapter 5 has been submitted to the following journal

- K. Achutegui and J. Míguez. "A Distributed particle filter for wireless sensor networks with stochastic observation exchange." submitted to Signal Processing.

6.2 Future research

6.2.1 Multiple targets

A natural extension of the single target tracking research, is the study and analysis of multiple target tracking.

Multi-target tracking consists of sequentially estimating the states of several targets from noisy data. It is encountered in many applications, e.g., aircraft tracking from radar measurements or football player tracking in a video sequence. Solutions of this problem using PFs have been proposed in the past ten years. Two problems are generally faced, namely the high dimension of the system (the state vector gathers all target states and its size increases with the number of targets) and the data association problem (often, each available measurement may correspond either to a target or to a false alarm and, conversely, each target may have been either detected or missed). An added difficulty is the modeling of the dynamics, including the possibility of a varying state dimension as targets appear and dissapear.

Classical multi-target tracking theory has focused on the second issue: the multiple hypothesis tracker (MHT) is based on filtering a set of detections or thresholded measurements using Kalman-filter-type algorithms, and the joint probabilistic data association (JPDA) filter is also classical approach to tackle the data association problem. Interestingly, this approach also solves the dimensional problem since it effectively resorts to one filter per target. For a small number of targets and measurements, this approach is very efficient. However, the number of possible associations combinatorially grows with the number of targets and measurements.

The independent partition particle filter (IPPF) considers multi-target tracking via particle filtering from a purely Bayesian perspective. Measurement-to-target association is not done explicitly; it is implicit within the Bayesian framework. Recently, a new approach to multi-target tracking has been developed based on random sets called finite-set statistics (FISST). FISST has also been combined with particle filtering methods for multi-target tracking.

To date, the implementation of MTT have been limited to small-scale problems for computational reasons and, thus, new methods are needed.

In order to handle the added computational complexity we propose the use of DPF schemes based on the DRNA algorithm in views of sharing the computational load among many nodes.

6.2.2 Convergence results

Another future line of research is to extend the convergence results published for the standard particle filter, e.g., the methodology of [32], to demonstrate the convergence of the DPFs based on the DRNA algorithm. The analysis should demonstrate what is the effect in the performance of the different steps of a DPF scheme in comparison with a CPF. For example, regarding

the particle exchange step, we should analyze how much precision is lost with it and what conditions are necessary in order to guarantee that the scheme works.

6.2.3 Suboptimal DPF schemes

The study of DPF schemes still allows for improvements. If we aim at reducing of the computational complexity, we can focus on the design of new suboptimal algorithms with more efficient importance functions. A possible efficient importance function could be an approximate optimal importance function obtained via diffusion methods. Another possibility would be to investigate the implementation of efficient particle filters, such as the auxiliary particle filter or the unscented particle filter, under the DRNA scheme.

On the other hand, if we aim at reducing the communication load, we could focus the research on new observation spread techniques, where, for example, we can prioritize the observations collected from sensors that are placed close to the target.

Appendix A

Recursive computation of the prior density

In order to compute the weights in Tables 3.2 and 3.3, we need to evaluate the predictive density of the position at time t, $p(\mathbf{r}_t|\mathbf{r}_{0:t-1}^{(i)}, \omega_{0:t-1}^{(i)})$. Given $\mathbf{r}_{0:t-1}^{(i)}$ and $\omega_{0:t-1}^{(i)}$, the dynamic model defined in (3.1) is linear and, therefore, the predictive density is Gaussian and can be obtained in closed form using a Kalman filter.

Consider the linear Gaussian state-space model

$$\underbrace{\begin{bmatrix} v_{1,t} \\ v_{2,t} \end{bmatrix}}_{\mathbf{v}_t} = \underbrace{\begin{bmatrix} \cos(\omega_{t-1}^{(i)}T) & -\sin(\omega_{t-1}^{(i)}T) \\ \sin(\omega_{t-1}^{(i)}T) & \cos(\omega_{t-1}^{(i)}T) \end{bmatrix}}_{\mathbf{H}_t^{(i)}} \underbrace{\begin{bmatrix} v_{1,t-1} \\ v_{2,t-1} \end{bmatrix}}_{\mathbf{v}_{t-1}} + T\mathbf{I}_2 \begin{bmatrix} u_{3,t} \\ u_{4,t} \end{bmatrix} \quad \text{(A.1)}$$

$$\underbrace{\begin{bmatrix} \Delta_{1,t} \\ \Delta_{2,t} \end{bmatrix}}_{\mathbf{\Delta}_t} = \underbrace{\begin{bmatrix} \frac{\sin(\omega_{t-1}^{(i)}T)}{\omega_{t-1}^{(i)}} & -\frac{\cos(\omega_{t-1}^{(i)}T)-1}{\omega_{t-1}^{(i)}} \\ \frac{1-\cos(\omega_{t-1}^{(i)}T)}{\omega_{t-1}^{(i)}} & \frac{\sin(\omega_{t-1}^{(i)}T)}{\omega_{t-1}^{(i)}} \end{bmatrix}}_{\mathbf{G}_t^{(i)}} \underbrace{\begin{bmatrix} v_{1,t-1} \\ v_{2,t-1} \end{bmatrix}}_{\mathbf{v}_{t-1}} + \frac{1}{2}T^2\mathbf{I}_2 \begin{bmatrix} u_{1,t} \\ u_{2,t} \end{bmatrix}, \quad \text{(A.2)}$$

where the velocity of the target, $\mathbf{v}_t$, is the system state, the changes in the position $\Delta_{1,t} = r_{1,t} - r_{1,t-1}^{(i)}$ and $\Delta_{1,t} = r_{2,t} - r_{2,t-1}^{(i)}$ are the observations and $u_{i,t} \sim N(u_{i,t}; 0, \sigma_u^2)$ is Gaussian noise, for $i = 1, \ldots, 4$.

If we define the matrices $\mathbf{W} = \sigma_u \begin{bmatrix} T & 0 \\ 0 & T \end{bmatrix}$ and $\mathbf{V} = \sigma_u \begin{bmatrix} \frac{1}{2}T^2 & 0 \\ 0 & \frac{1}{2}T^2 \end{bmatrix}$, then we can recursively apply the Kalman filter [27] to the model of (A.1)

and (A.2) at each time instant, to obtain

$$\begin{aligned}
\mathbf{P}_t^{(i)} &= \mathbf{H}_{t-1}^{(i)} \boldsymbol{\Sigma}_{t-1}^{(i)} {\mathbf{H}_{t-1}^{(i)}}^{\top} + \mathbf{W}\mathbf{W}^{\top}, \\
\mathbf{S}_t^{(i)} &= \mathbf{G}_t^{(i)} \mathbf{P}_t^{(i)} {\mathbf{G}_t^{(i)}}^{\top} + \mathbf{V}\mathbf{V}^{\top}, \\
\boldsymbol{\mu}_t^{(i)} &= \mathbf{H}_t^{(i)} \boldsymbol{\mu}_{t-1}^{(i)} + \mathbf{P}_t^{(i)} {\mathbf{G}_t^{(i)}}^{\top} {\mathbf{S}_t^{(i)}}^{-1} \left(\boldsymbol{\Delta}_t - \mathbf{G}_t^{(i)} \mathbf{H}_t^{(i)} \boldsymbol{\mu}_{t-1}^{(i)} \right), \\
\boldsymbol{\Sigma}_t^{(i)} &= \mathbf{P}_t^{(i)} - \mathbf{P}_t^{(i)} {\mathbf{G}_t^{(i)}}^{\top} {\mathbf{S}_t^{(i)}}^{-1} \mathbf{G}_t^{(i)} \mathbf{P}_t^{(i)}, \qquad \text{(A.3)}
\end{aligned}$$

where $\mathbf{P}_t^{(i)}$ is the prior covariance matrix of $\mathbf{v}_t$; $\mathbf{S}_t^{(i)}$ is the covariance of $\boldsymbol{\Delta}_t$ given $\boldsymbol{\Delta}_{0:t-1}^{(i)}$; $\boldsymbol{\mu}_t^{(i)}$ is the posterior mean of $\mathbf{v}_t$ given $\mathbf{r}_{0:t}^{(i)}$ and $\boldsymbol{\Sigma}_t^{(i)}$ is the posterior covariance of $\mathbf{v}_t$ given $\mathbf{r}_{0:t}^{(i)}$.
In addition, the predictive distribution of $\boldsymbol{\Delta}_t$ is

$$p(\boldsymbol{\Delta}_t | \boldsymbol{\Delta}_{0:t-1}^{(i)}) \sim N(\boldsymbol{\Delta}_t; \mathbf{G}_t^{(i)} \mathbf{H}_t^{(i)} \boldsymbol{\mu}_{t-1}^{(i)}, \mathbf{S}_t^{(i)}). \qquad \text{(A.4)}$$

Therefore, we can obtain the prior distribution for the position, $\mathbf{r}_t$, as

$$p(\mathbf{r}_t | \mathbf{r}_{0:t-1}^{(i)}, \omega_{0:t-1}^{(i)}, a_{0:t-1}^{(i)}) \sim N(\mathbf{r}_t; \overline{\mathbf{r}}_{t|t-1}^{(i)}, \overline{\boldsymbol{\Sigma}}_{t|t-1}^{(i)}) \qquad \text{(A.5)}$$

where $\overline{\mathbf{r}}_{t|t-1}^{(i)} = \mathbf{G}_t^{(i)} \mathbf{H}_t^{(i)} \boldsymbol{\mu}_{t-1}^{(i)} + \mathbf{r}_{t-1}^{(i)}$ and $\overline{\boldsymbol{\Sigma}}_{t|t-1}^{(i)} = \mathbf{S}_t^{(i)}$.

Appendix B

Derivation of the likelihood

Since the model indices $m_{1,t}, \ldots, m_{J,t}$ are statistically independent (as assumed in Section 3.2.2) Eq.(5.2) implies that the observations $y_{1,t}, \ldots, y_{J,t}$ are conditionally independent given the target position $\mathbf{r}_t$, i.e.

$$p(\mathbf{y}_t|\mathbf{r}_t) = \prod_{j=1}^{J} p(y_{j,t}|\mathbf{r}_t). \tag{B.1}$$

Each likelihood factor $p(y_{j,t}|\mathbf{r}_t)$ can be written as a marginal of the joint probability function $p(y_{j,t}, m_{j,t}|\mathbf{r}_t)$,

$$p(y_{j,t}|\mathbf{r}_t) = \sum_{m_{j,t}=1}^{K} p(y_{j,t}, m_{j,t}|\mathbf{r}_t)p(m_{j,t}), \tag{B.2}$$

where we have used the independence of $\mathbf{r}_t$ and $m_{j,t}$. Finally substituting (B.2) into (B.1) yields

$$p(\mathbf{y}_t|\mathbf{r}_t) = \prod_{j=1}^{J} \sum_{m_{j,t}=1}^{K} p(y_{j,t}|\mathbf{r}_t, m_{j,t})p(m_{j,t}). \tag{B.3}$$

Appendix C

Acronyms and abbreviations

- AoA: angle of arrival.
- APF: auxiliary particle filter.
- A-RBPF: auxiliary Rao-Blackwellized particle filtering.
- A-SIR. auxiliary sequential importance resampling.
- CPF: centralized particle filter.
- CPU: central processing unit.
- CT: coordinated turn.
- CV: constant velocity.
- DFT: discrete Fourier transform.
- DPF: distributed particle filter.
- DRNA: distributed resampling with non-proportional allocation.
- e.g.: exempli gratia (for instance).
- EKF: extended Kalman filter.
- fps: false positive rate.
- fnr: false negative rate.
- GPS: global positioning system.

- GSMM: generalized switching multiple model.
- i.e.: id est (that is).
- i.i.d.: independent and identically distributed.
- IMM: interacting multiple model.
- IMM-UKF: interacting multiple model unscented Kalman filter.
- IS: importance sampling.
- JMS: jump Markov system.
- KF: Kalman filter.
- LS: least squares.
- MAE: mean absolute error.
- MAP: maximum a posteriori.
- MKF: mixture Kalman filter.
- ML: maximum likelihood.
- MMSE: minimum mean-square error.
- NLS: nonlinear least squares.
- OOsM: out-of-sequence measurement.
- PAN: personal area network.
- PE: processing element.
- PF: particle filter.
- pdf: probability density function.
- RBPF: Rao-Blackwellized particle filtering.
- RFID: radio frequency identification.
- RSS: received signal strength.
- r.v.: random variable.

- RLS: recursive least squares.
- SDE: standard deviation of error.
- SE: sensing element.
- SIR: sequential importance resampling.
- SIS: sequential importance sampling.
- SIS/R: sequential importance sampling with resampling.
- SLS: separable least squares.
- SMC: sequential Monte Carlo.
- SMM: switching multiple models.
- TDoA: time difference of arrival.
- ToA: time of arrival.
- UKF: unscented Kalman filter.
- WLS: weighted least squares.
- WSN: wireless sensor networks.

Appendix D

Notation

- x: scalar magnitudes are denoted using lower case regular face letters.
- $\mathbf{x}$: vectors are displayed as lower case bold-face letters.
- $\mathbf{X}$: matrices are displayed as upper case bold-face letters.
- $\mathbf{x} = [x_1, ..., x_n]$: the scalar coordinates of a row vector in n-dimensional space are denoted with square brackets.
- $\mathbf{x} = [x_1, ..., x_n]^\top$: a column vector is described a the transpose of a row vector.
- x, $\mathbf{x}$ or $\mathbf{X}$: random variables are denoted using upper or lower case and regular or bold face letters according to their dimension.
- $x \in \mathcal{R}$: sample space of random variable x is the set of real numbers.
- $\mathbf{x} \in \mathcal{R}^2$: random variable $\mathbf{x}$ is of dimension 2 and its sample space is the set of real numbers.
- $x \sim p(x)$: means that a random variable or a sample x has the indicated distribution.
- $p(\cdot)$: (lower case letter) probability density function (pdf) of a random variable or vector.
- $p(x|y)$: the conditional pdf of x given y.
- Prob$\{\cdot\}$: the probability of an event.
- $\mathcal{U}([a, b])$: uniform distribution in the interval between a and b.

- $\mathcal{N}(\mu, \sigma^2)$: normal distribution with mean μ and variance σ^2.
- $\mathcal{N}(x; \mu, \sigma^2)$: evaluation of the normal pdf with mean μ and variance σ^2 in x.
- $\{x^{(i)}\}_{i=1}^{N}$: set of N samples.
- $x^{(i)}$: i-th sample of a set $\{x\}_{i=1}^{N}$.
- x_t: sample related to time instant t.
- $x_t^{(i)}$: i-th sample related to time instant t.

Bibliography

[1] K. Achutegui, L. Martino, J. Rodas, C.J. Escudero, and J. Miguez. A multi-model particle filtering algorithm for indoor tracking of mobile terminals using RSS data. *IEEE Control Applications, (CCA) and Intelligent Control, (ISIC)*, 2009.

[2] Nadeem Ahmed, Yifei Dong, tatiana Bokareva, Salil Kanhere, Sanjay Jha, Travis Bessell, Mark Rutten, Branko Ristic, and Neil Gordon. Detection and tracking using wireless sensor networks. In *5th international conference on Embedded networked sensor systems*, SenSys '07, pages 425–426. ACM, 2007.

[3] B. D. O. Anderson and J. B. Moore. *Optimal Filtering*. Englewood Cliffs, 1979.

[4] C. Andrieu, A. Doucet, S. S. Singh, and V. B. Tadić. Particle Methods for Change Detection, System Identification and Control. *Proceedings of the IEEE*, 92(3):423–438, March 2004.

[5] Pau Closas Anup Dhital and Carles Fernández-Prades. Bayesian filtering for indoor localization and tracking in wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2012.

[6] M. Arikawa, S. Konomi, and K. Ohnishi. Navitime: Supporting Pedestrian Navigation in the Real World. *IEEE Pervasive Computing*, 6(3):21–29, 2007.

[7] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Klapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions Signal Processing*, 50(2):174–188, February 2002.

[8] A.H. Sayed B. Hassibi and T. Kailath. Linear estimation in Krein spaces-Part II: applications. *IEEE Transactions on Automatic Control*, 41(1):34–49, 1996.

[9] P. Bahl and V. Padmanabhan. Radar: An In-Building RF Based User Location and Tracking System. *IEEE Computer and Communications Societies (INFOCOM 2000)*, pages 775–784, March 2000.

[10] Alan M. Bain and Dan Crisan. *Fundamentals of stochastic filtering*. Springer, 2008.

[11] Y. Bar-Shalom and X. R. Li, editors. *Estimation with Applications to Tracking and Navigation*. Wiley & sons, 2001.

[12] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[13] H. A. P. Blom and E. A. Bloem. Exact Bayesian and particle filtering of stochastic hybrid systems. *IEEE Transactions Aerospace and Electronic Systems*, 43(1):55–70, 2007.

[14] M. Bolić, P. M. Djurić, and S. Hong. New Resampling Algorithms for Particle Filters. In *Proceedings of the IEEE ICASSP*, April 2003.

[15] M. Bolić, P. M. Djurić, and S. Hong. Resampling Algorithms and Architectures for Distributed Particle Filters. *IEEE Transactions Signal Processing*, 53(7):2442–2450, July 2005.

[16] Claudio J. Bordin and Marcelo G. S. Bruno. Cooperative Bling Equalization of Frequency-Selective Channels in Sensor Networks using Decentralized Particle Filtering. *42nd Asilomar Conference on Signals, systems and computers*, pages 1198 – 1201, October 2008.

[17] D. C. Brogan and N. L. Jhonson. Realistic Human Walking Paths. *Proceedings of CASA, International Conference on Computer Animation and Social Agents*, May 2003.

[18] M. Brunato and R. Battiti. Statistical learning theory for location fingerprinting in wireless LANs. *Computer Networks*, 47:825–845, April 2005.

[19] E. Bruns, B. Brombach, T. Zeidler, and O. Bimber. Enabling mobile phones to support large-scale museum guidance. *IEEE Multimedia*, 14(2):16–25, 2007.

[20] M. F. Bugallo, S. Xu, and P. M. Djurić. Performance comparison of EKF and particle filtering methods for maneuvering targets. *Digital Signal Processing*, 17:774–786, October 2007.

[21] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low-cost outdoor localization for very small devices. *Personal Communications, IEEE*, 7(5):28–34, 2000.

[22] Olivier Cappé, Simon J Godsill, and Eric Moulines. An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE*, 95(5):899–924, 2007.

[23] F. Caron, M. Davy, E. Duflos, and F. Vanheeghe. Particle Filtering for Multisensor Data Fusion with Switching Observation Models: Application to Land Vehicle Positioning. *IEEE Transactions Signal Processing*, 55(6):2703–2719, June 2007.

[24] J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for nonlinear problems. *IEE Proceedings - Radar, Sonar and Navigation*, 146(1):2–7, February 1999.

[25] G. Casella and C. P. Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

[26] Federico S. Cattivelli and Ali H. Sayed. Diffusion LMS strategies for distributed estimation. *Trans. Sig. Proc.*, 58(3):1035–1048, March 2010.

[27] R. Chen and J. S. Liu. Mixture Kalman filters. *Journal of the Royal Statistics Society B*, 62:493–508, 2000.

[28] Daizhan Cheng, Bijoy Ghosh, and Xiaoming Hu. Distributed Sensor Network for Target Tracking. In *17th International Symposium on Mathematical Theory of Networks and Systems*, 2006.

[29] P. Closas and M.F. Bugallo. Iterated multiple particle filtering. In *4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 89–92, December 2011.

[30] M. Coates. Distributed particle filters for sensor networks. In *Proceedings of ACM IPSN*, April 2004.

[31] D. Crisan. Particle Filters - A Theoretical Perspective. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, chapter 2, pages 17–42. Springer, 2001.

[32] D. Crisan and A. Doucet. A Survey of Convergence Results on Particle Filtering. *IEEE Transactions Signal Processing*, 50(3):736–746, March 2002.

[33] Digant P. Davé and Thomas E. Milner. Doppler-angle measurement in highly scattering media. *Opt. Lett.*, 25(20):1523–1525, Oct 2000.

[34] A. Dhital, P. Closas, and C. Fernández-Prades. Bayesian filtering for indoor localization and tracking in wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2012.

[35] P. M. Djurić, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Míguez. Particle filtering. *IEEE Signal Processing Magazine*, 20(5):19–38, September 2003.

[36] P. M. Djurić, Ting. Lu, and M. F. Bugallo. Multiple particle filtering. In *32nd IEEE ICASSP*, April 2007.

[37] P. M. Djuric, M. Vemula, and M. F. Bugallo. Target tracking by particle filtering in binary sensor networks. *IEEE Transactions on Signal Processing*, 56:2229–2238, June 2008.

[38] R. Douc, O. Cappé, and E. Moulines. Comparison of resampling schemes for particle filtering. In *Proceedings of the 4^{th} International Symposium on Image and Signal Processing and Analysis*, pages 64–69, September 2005.

[39] A. Doucet. On sequential simulation-based methods for Bayesian filtering. *Tech. Report, Univ. of Cambridge, Dept. of Engineering, CUED-F-ENG-TR310*, 1998.

[40] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, New York (USA), 2001.

[41] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo Sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[42] D.P. Eickstedt and M. R. Benjamin. Cooperative target tracking in a distributed autonomous sensor network. In *OCEANS 2006*, pages 1–6, September 2006.

[43] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: scalable coordination in sensor

networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '99, pages 263–270, New York, NY, USA, 1999. ACM.

[44] D. Fox, Jeffrey Hightower, Lin Liao, D. Schulz, and G. Borriello. Bayesian filtering for location estimation. *Pervasive Computing, IEEE*, 2(3):24–33, 2003.

[45] S. Godsill, A. Doucet, and M. West. Monte Carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, 99(465):156–168, March 2004.

[46] G. Goncalo and S. Helena. Indoor location system using ZigBee technology. In *Sensor Technologies and Applications, 2009. SENSORCOMM '09. Third International Conference on*, pages 152–157, 2009.

[47] N. Gordon, D. Salmond, and A. F. M. Smith. Novel approach to nonlinear and non-Gaussian Bayesian state estimation. *IEE Proceedings-F*, 140:107–113, 1993.

[48] F. Gustafsson. *Statistical Sensor Fusion*. Studentlitteratur, 2010.

[49] F. Gustafsson and F. Gunnarsson. Mobile positioning using wireless networks. *IEEE Signal Processing Magazine*, 22(4):41–53, July 2005.

[50] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation and tracking. *IEEE Transactions Signal Processing*, 50(2):425–437, February 2002.

[51] M. Haller, M. Billinghurst, and B. Thomas. *Emerging Technologies of Augmented Reality: Interfaces and Design*. IGI Global, 2007.

[52] M.A. Hanson, H.C. Powell, A.T. Barth, K. Ringgenberg, B.H. Calhoun, J.H. Aylor, and J. Lach. Body area sensor networks: Challenges and opportunities. *Computer*, 42(1):58–65, 2009.

[53] Y. Hatano and M. Mesbahi. Agreement over random networks. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 2, pages 2010–2015 Vol.2, 2004.

[54] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks.

In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, page 10 pp. vol.2, 2000.

[55] O. Hlinka, P.M. Djuric, and F. Hlawatsch. Time-space-sequential distributed particle filtering with low-rate communications. In *Signals, Systems and Computers, 2009 Conference Record of the Forty-Third Asilomar Conference on*, pages 196–200, 2009.

[56] O. Hlinka, O. Sluciak, F. Hlawatsch, P. Djuric, and M. Rupp. Distributed Gaussian particle filtering using likelihood consensus. In *Proceedings of IEEE ICASSP*, pages 3756–3759, May 2011.

[57] O. Hlinka, O. Sluciak, F. Hlawatsch, P. M. Djuric, and M. Rupp. Likelihood Consensus and Its Application to Distributed Particle Filtering. *IEEE Transactions on Signal Processing*, 60:4334–4349, August 2012.

[58] Heng-Chih Huang, Yueh-Min Huang, and Jen-Wen Ding. An implementation of battery-aware wireless sensor network using ZigBee for multimedia service. In *Consumer Electronics, 2006. ICCE '06. 2006 Digest of Technical Papers. International Conference on*, pages 369–370, 2006.

[59] Y. Sankarasubramanian I. F. Akyildiz, W-Su and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 38(4):393–422, March 2002.

[60] Garrick Ing and Mark J. Coates. Parallel particle filters for tracking in wireless sensor networks. In *in Proc. SPAWC*, 2005.

[61] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom '00, pages 56–67, New York, NY, USA, 2000. ACM.

[62] R. Jan and Y. Lee. An indoor geolocation system for wireless LANs. *International Conference on Parallel Processing Workshops*, pages 29–34, October 2003.

[63] L.A. Jhonston and V. Krishnamurthy. An improvement to the interacting multiple model (IMM) algorithm. *IEEE Transactions on Signal Processing*, 46(2):2909–2903, 2001.

[64] B. Jit, D. Zhang, G. Qiao, V. Foo, Q. Qiu, and P. Yap. A system for activity monitoring and patient tracking in a smart hospital. *International Conference on Smart Homes and Health Telematics (ICOST'06)*, pages 196–263, 2006.

[65] S. J. Julier and J. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(2):401–422, March 2004.

[66] Simon J. Julier and Jeffrey K. Uhlmann. New extension of the Kalman filter to nonlinear systems. In *Proc. AeroSense: 11th Int. Symp. Aerospace/Defense Sensing, Simulation and Controls*, pages 182–193, 1997.

[67] K. Kaemarungsi. Distribution of WLAN received signal strength indication for indoor location determination. *International Symposium on Wireless Pervasive Computing*, pages 6–11, January 2006.

[68] S. Kar and J.M.F. Moura. Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise. *Signal Processing, IEEE Transactions on*, 57(1):355–369, 2009.

[69] Rebecca Adam Kathrin Schmeink and Peter A Hoeher. Joint communication and positioning based on soft channel parameter estimation. *EURASIP Journal on Wireless Communications and Networking*, November 2011.

[70] C. J. Kim and C.R. Nelson. *State-Space Models With Regime Switching*. MIT Press, 1999.

[71] N. Kumar. *Comprehensive Physics XII*. Laxmi Publications, 2008.

[72] H. R. Künsch. Recursive Monte Carlo filters: Algorithms and theoretical bounds. *The Annals of Statistics*, 33(5):1983–2021, 2005.

[73] A.M. Ladd, K.E. Bekris, A.P. Rudys, D.S. Wallach, and E.E. Kavraki. On the feasibility of using wireless ethernet for indoor localization. *Robotics and Automation, IEEE Transactions on*, 20(3):555–559, 2004.

[74] Koen Langendoen and Niels Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Elsevier Computer Networks*, 43:499–518, November 2003.

[75] A. Lee, C. Yau, M. B. Giles, A. Doucet, and C. C. Holmes. On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *Journal of Computational and Graphical Statistics*, 19(4):769–789, 2010.

[76] Sun Hwan Lee and Matthew West. Markov Chain Distributed Particle Filters (MCDPF). *48th Conference on Decision and Control and 28th Chinese Control Conference*, pages 5496–5501, December 2009.

[77] J. S. Liu, R. Chen, and W. H. Wong. Rejection control and sequential importance sampling. *Journal of the American Statistical Association*, 93(443):1022–1031, September 1998.

[78] T. Kirubarajan M. Mallick and S. Arulampalam. Out-of-sequence measurements processing for tracking ground target using particle filters. In *Proceedings of IEEE Aerospace Conference*, volume 41, pages 1809–1818, 2002.

[79] E. Mazor, A. Averbuch, and Y. Bar-Shalom. Interacting multiple model methods in target tracking: A survey. *IEEE Transactions Aerospace and Electronic Systems*, 34(1):103–123, 1998.

[80] S. McGinnity and G. W. Irwin. Manoeuvring target tracking using a multiple-model bootstrap filter. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, chapter 23, pages 479–496. Springer, 2001.

[81] J. Míguez. Analysis of parallelizable resampling algorithms for particle filtering. *Signal Processing*, 87(12):3155–3174, 2007.

[82] Aleksandar Milenkovi, Chris Otto, and Emil Jovanov. Wireless sensor networks for personal health monitoring: Issues and an implementation. *Computer Communications*, 29(13-44):2521 – 2533, 2006. Wirelsess Senson Networks and Wired/Wireless Internet Communications.

[83] Erik G Strm Mohammad R Gholami, Henk Wymeersch and Mats Rydstrm. Wireless network positioning as a convex feasibility problem. *Eurasip Journal on Wireless Communications and Networking*, November 2011.

[84] E. J. Msechu, A. Ribeiro, S. I. Roumeliotis, and G. B. Giannakis. Distributed iteratively quantized Kalman filtering for wireless sensor

networks. *IEEE Transactions Signal Processing*, 56:3727–3741, August 2008.

[85] R. Musaloiu and A. Terzis. Minimising the effect of WiFi interference in 802.15.4 wireless sensor networks. *International Journal of Sensor Networks*, 3(1):43–54, 2008.

[86] M. O'Connor, T. Bell, G. Elkaim, and B. Parkinson. Automatic steering of farm vehicles using GPS. *3rd International Conference on Precision Agriculture*, pages 767–778, June 1996.

[87] R. Olfati-Saber. Distributed Kalman filtering for sensor networks. In *46th IEEE Conference on Decision and Control*, pages 5492 – 5498, December 2007.

[88] Eric A. Olsen, Chan-Woo Park, and Jonathan P. How. 3D formation flight using differential carrier-phase GPS sensors, 1998.

[89] U. Orguner and F. Gustafsson. Storage efficient particle filters for the out of sequence measurement problem. *11th International Conference on Information Fusion*, pages 1–8, July 2008.

[90] M. Orton and A. Marrs. Particle filters for tracking with out-of-sequence measurements. *IEEE Transactions on Aerospace and Electronic Systems*, 41:693–702, April 2005.

[91] Matthew Orton and Alan Marrs. A Bayesian approach to multi-target tracking and data fusion with out-of-sequence measurements. *Proceedings of IEE Conference on Target Tracking: Algorithms and Applications*, 1:1–5, 2001.

[92] Cheolhee Park and T.S. Rappaport. Short-range wireless communications for next-generation networks: UWB, 60 Ghz millimeter-wave WPAN, and ZigBee. *Wireless Communications, IEEE*, 14(4):70–78, 2007.

[93] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero III, R. L. Moses, and N. S. Correal. Locating the nodes. *IEEE Signal Processing Magazine*, 22(4):54–69, July 2005.

[94] M. K. Pitt and N. Shephard. Auxiliary variable based particle filters. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, chapter 13, pages 273–293. Springer, 2001.

[95] T. Preis, P. Virnau, W. Paul, and J. J. Schneider. GPU accelerated Monte Carlo simulation of the 2D and 3D Ising model. *Journal of Computational Physics*, 228(12):4468–4477, 2009.

[96] H. Qi and F. Wang. Optimal itinerary analysis for mobile agents in ad hoc wireless sensor networks. In *Proceedings of IEEE ICC 01*, pages 147 –153, 2001.

[97] T. S. Rappaport. *Wireless Communications*. Prentice-Hall, Upper Saddle River, NJ (USA), 1996.

[98] T. S. Rappaport. *Wireless Communications: Principles and Practice (2nd edition)*. Prentice-Hall, Upper Saddle River, NJ (USA), 2001.

[99] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter*. Artech House, Boston, 2004.

[100] Sheldon M Ross. *Applied probability models with optimization applications*. Courier Dover Publications, 1970.

[101] D. B. Rubin. A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: the SIR algorithm. *Journal of the American Statistical Association*, 82:543–546, 1987.

[102] J. Uhlmann S. J. Julier and H. F. Durrant-Whyte. A new method for the non linear transformation of means and covariances in filters and estimators. *IEEE Transactions Automatic Control*, 3:477–482, March 2000.

[103] C. Savarese, J.M. Rabaey, and J. Beutel. Location in distributed ad-hoc wireless sensor networks. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, volume 4, pages 2037–2040 vol.4, 2001.

[104] Robert J. Schalkoff. *Pattern Recognition*. John Wiley & Sons, Inc., New York, 1991.

[105] X. Sheng, Yu-Hen Hu, and P. Ramanathan. Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network. In *Fourth International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 181– 188, April 2005.

[106] M. L. Sichitu and V. Ramadurai. Locatino of wireless sensor networks with a mobile beacon. In *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 174– 183, October 2004.

[107] F. Simjee and P.H. Chou. Everlast: Long-life, supercapacitor-operated wireless sensor node. In *Low Power Electronics and Design, 2006. ISLPED'06. Proceedings of the 2006 International Symposium on*, pages 197–202, 2006.

[108] R. Simon, P. Frohlich, and H. Anegg. Beyond location based - the spatially aware mobile phone. *Web and Wireless Geographical Information Systems*, pages 12–21, 2006.

[109] M. A. Suchard, Q. Wang, C. Chang, J. Frelinger, A. Cron, and M. West. Understanding GPU programming for statistical computation: Studies in massively parallel massive mixtures. *Journal of Computational and Graphical Statistics*, 19(4):419–438, 2010.

[110] D. D. Sworder and J. Boyd. *Estimation Problems in Hybrid Systems*. Cambdridge University Press, 1999.

[111] Su-Won Yoon Tae-Hong Shin, Sangyoon Chin and Soon-Wook Kwon. A service-oriented integrated information framework for RFID/WSN-based intelligent construction supply chain management. *Elsevier Automation in Construction*, 20(20):706–715, October 2011.

[112] J. Lygeros Th. Arampatzis and S. Manesis. A Survey of Applications of Wireless Sensors and Wireless Sensor Networks. In *Proceedings of the 13th Mediterranean Conference on Control and Automation*, pages 719–724, June 2005.

[113] F. Viani, P. Rocca, G. Oliveri, D. Trinchero, and A. Massa. Localization, tracking, and imaging of targets in wireless sensor networks: An invited review. *Radio Science*, 46(5):n/a–n/a, 2011.

[114] Eric A. Wan and Rudolph van der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE Symposium on Adaptive System, Signal Processing and Communication Control*, October 2000.

[115] R. Weinstein. RFID: a technical overview and its application to enterprise). *IT Proffesional, Sponsored by IEEE Computer Society*, 7:27–33, June 2005.

[116] G. Welch and G. Bishop. *An introduction to the Kalman filter*. UNC-Chapel Hill, http://www.cs.unc.edu, 2000.

[117] X.R. Li Y. Bar-Shalom. *Estimation and Tracking Principles, Techniques and Software*. Artech House, 1993.

[118] Y. Bar-Shalom y W. D. Blair, editor. *Multitarget-multisensor tracking: Applications and advances. Volume III*. Artech House, Norwood (MA, USA), 2000.

[119] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292 – 2330, 2008.